НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

«На правах рукопису»
УДК 004.89

«До захисту допущено»
Завідувач кафедри
_____ Коваль О. В.
(підпис) (ініціали, прізвище)
“___”_____20__ р

Магістерська дисертація

зі спеціальності **121 Інженерія програмного забезпечення**
за спеціалізацією Інженерія програмного забезпечення розподілених систем
на тему: Автоматичне виділення концептів та тез в онтологічно-орієнтованому навчальному порталі
Виконав: студент 6 курсу, групи ТВ-82мп
(шифр групи)

Суходольський Артем Олексійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., Титенко С. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет **Теплоенергетичний**
(повна назва)

Кафедра **Автоматизації проектування енергетичних процесів і систем**
(повна назва)

Рівень вищої освіти другий (магістерський) за освітньо-професійною програмою

Спеціальніст **121 Інженерія програмного забезпечення**
(код і назва)

Спеціалізаці **Інженерія програмного забезпечення розподілених систем**

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Коваль О. В.
(підпис) (ініціали, прізвище)

“___” _____20__ р

ЗАВДАННЯ

на магістерську дисертацію студенту

Суходольському Артему Олексійовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації “Автоматичне виділення концептів та тез в онтологічно-орієнтованому навчальному порталі”

Науковий керівник Титенко Сергій Володимирович, к.т.н. , доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджено наказом по університету 3812-с від “4” листопада 2019 р.

2. Строк подання студентом дисертації 12 грудня 2019 р.

3. Об’єкт дослідження: Сервіс автоматичного пошуку концептів та тез

4. Предмет дослідження: Методи для автоматичного пошуку концептів та тез

5. Перелік завдань, які необхідно розробити:

1) Проаналізувати методи автоматичного пошуку концептів та тез
2) Проаналізувати технологію генерації завдань на базі ПТМ

3) Проаналізувати ряд методів оптимізації ПТМ
4) Розробити метод генерації тестів з диференціацією за складністю
5) Розробити сервіс для автоматизованого тестування

6. Орієнтований перелік ілюстративного матеріалу:
1) Use case діаграма в нотації UML
2) DFD діаграма роботи програмного комплексу
3) Приклади роботи з веб-інтерфейсом

7. Орієнтований перелік публікацій:
Search index building for numeric vectors with O(1) memory consumption (збірник "Modern aspects of software development" Proceedings of VI International Scientific and Practical Virtual Conference of Software Development Specialists)

8. Дата видачі завдання: "28" вересня 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	28.09.19р.	
2	Опрацювання літературних джерел	01.10.18 р. – 03.02.19р.	
3	Підготовка матеріалів дисертації	04.02 – 31.05.19 р.	
4	Підготовка доповідей на конференції	01.05 – 20.05.19 р.	
5	Розробка програмного продукту	03.06 – 25.10.19 р.	
6	Переддипломна практика	2.09 – 25.10.19 р.	
7	Захист програмного продукту	25.10.19р.	
8	Розробка стартап-проекту	11.11 – 19.11.19 р.	
9	Передзахист	22.11.19 р.	
10	Оформлення дисертації	21.11- 29.11.19 р.	
11	Захист	18.12.19 р.	

Студент _____ Суходольський А.О.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Титенко С. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дисертаційна робота складається зі вступу, 5 розділів, висновків, списку використаних джерел з 30 найменувань. Обсяг дисертації становить 75 сторінок, робота має 12 рисунків, 18 таблиць, 8 формул та 1 додаток.

Актуальність теми. У сучасному світі, текстові дані займають значну частку в інформаційних системах, у тому числі у системах дистанційної освіти, де якісна структуризація знань напряму корелює з якістю всього сервісу. Класичними методами являються збереження даних у базах даних у вигляді тексту та тегів для нього, щоб підкреслити ієрархію статей та структурувати їх. Одним із підходів у якісному наданні знань являється створення понятійно-тезисної моделі (ПТМ), яка містить в собі інформацію о концептах та тезах до них. ПТМ надає засоби опису понятійної складової контенту і забезпечує основу для програмного інструментарію редагування і використання бази даних та знань. Оскільки ПТМ являє собою складну структуру представлення даних, то її наповнення супроводжується ручною роботою, що може негативно впливати на ефективність її створення.

Тому постає задача у створенні методу для автоматичного виділення концептів та тез, який буде працювати на базі статей з онтологічно-орієнтованого порталу. Даний метод повинен надавати користувачу якісні рекомендації концептів та тез, які відповідають змісту статті. Також є необхідність у створенні сервісу, який дозволяє іншим веб-застосункам використовувати можливості сервісу для вирішення своїх задач у контексті створення ПТМ.

Мета та завдання дослідження. Метою даної дисертаційної роботи є створення формального апарату та програмного продукту, який автоматизує процес виділення концептів та тез на основі текстових даних на англійській мові. Також метою являється аналіз сучасних програмних продуктів та алгоритмів, які виконують

поставлену задачу, зіставити їх з реаліями освітнього порталу, та, при необхідності, адаптувати зі збереженням якості їх роботи.

Поставлена мета досягається шляхом вирішення наступних завдань:

- аналіз чинних алгоритмічних рішень для вирішення проблеми автоматичного виділення концептів та тез;
- розробка формального апарату для методу автоматичного виділення концептів та тез за тексту на англійській мові;
- розробка та програмна реалізація системи для автоматичного виділення концептів та тез, зокрема створення REST API інтерфейсу та веб-інтерфейсу користувача.

Об’єкт дослідження. Сервіс автоматичного пошуку концептів та тез

Предмет дослідження. Методи для автоматичного пошуку концептів та тез

Методи дослідження. Методи досліджень включають в себе створення метрики для вимірювання якості виділених концептів. Дана метрика включає в себе вручну оброблені дані (список концептів для статті). Також був використаний метод імітації моделювання навантаженості сервісу, де система опрацьовувала велику кількість запитів.

Інноваційна новизна одержаних результатів полягає у вирішенні актуальної проблеми автоматичного виділення понять та тез, а саме:

- запропоновано формальний апарат для виділення концептів на базі запропонованої мови регулярних виразів, їх фільтрації та ранжування. Також був запропонований метод для автоматичного пошуку відповідних тез для концептів;
- адаптовані методи для токенізації тексту та класифікації слів на частини мови.

Практичне значення отриманих результатів полягає в розробці програмної системи для автоматичного виділення концептів та тез та їх рекомендація експерту з онтології.

Апробація результатів дисертації. Результати досліджень, включених до дисертації, представлені на VI міжнародній науково-практичній віртуальній конференції фахівців з розробки програмного забезпечення на тему “Сучасні аспекти розробки програмного забезпечення” 2019-го року.

Публікації. Тези доповідей публікувалися в збірнику “Сучасні аспекти розробки програмного забезпечення” VI міжнародної науково-практичної віртуальної конференції фахівців з розробки програмного забезпечення.

Ключові слова. АВТОМАТИЧНЕ ВИДІЛЕННЯ КОНЦЕПТІВ, ОНТОЛОГІЯ, ПОНЯТІЙНО-ТЕЗИСНА МОДЕЛЬ, АВТОМАТИЧНА ОБРОБКА ТЕКСТІВ.

ABSTRACT

This dissertation consists of an introduction, 5 sections, conclusions, a list of used sources of 30 items. The dissertation size is 75 pages, the work has 12 drawings, 18 tables, 8 formulas and 1 appendix.

Actuality of theme. In modern world, textual data has a significant share in information systems, including distance education, where the qualitative structuring of knowledge directly correlates with the quality of the entire service. The classic methods are to store data in databases as text and tags for it, to emphasize the hierarchy of articles and to structure them. One of the approaches to quality knowledge delivery is to create a conceptual thesis model (CTM), which contains information about concepts and theses to them. CTM provides a means of describing the conceptual content component and provides the basis for software tools for editing and using the database and knowledge. Since CTM is a complex structure for data representation, filling it is accompanied by manual work, which can negatively affect the efficiency of its creation.

Therefore, the task is to create a method for the automatic selection of concepts and theses, which will work on the basis of articles from the ontology-oriented portal. This method should provide the user with quality guidelines for concepts and theses that are relevant to the content of the article. There is also a need to create a service that allows other web applications to use the service capabilities to solve their tasks in the context of creating a CTM.

The purpose and objectives of the study. The purpose of this dissertation is to create a formal apparatus and software that automates the process of highlighting concepts and theses based on textual data in English. The purpose is also to analyze modern software products and algorithms that perform the task, to compare them with the realities of the educational portal, and, if necessary, to adapt to preserve the quality of their work.

This goal is achieved by solving the following problems:

- analysis of existing algorithmic solutions to solve the problem of automatic extraction of concepts and theses;
- development of a formal apparatus for the method of automatic extraction of concepts and theses in English;
- development and software implementation of the system for automatic concepts and theses extraction including creation of REST API an user web-interface.

Object of study. Automatic search for concepts and theses.

Subject of study. Methods for automatic search for concepts and theses.

Research methods. Research methods include creating a metric to measure the quality of selected concepts. This metric includes manually processed data. We also simulated service workload, where the system processed a large number of requests.

The innovative novelty of the obtained results is to solve the urgent problem of automatic separation of concepts and abstracts, namely:

- a formal apparatus for concepts extraction based on the proposed regular expression language, filtering and ranking them. A method was also proposed to automatically find relevant theses for concepts;
- adapted methods for text tokenization and part of speech tagging.

The practical significance of the obtained results lies in the development of a software system for the automatic selection of concepts and theses and their recommendation to an ontology expert.

Testing the results of this work. The results of the studies included in dissertations presented at VI International Virtual Practical Virtual conferences of experts on software development on the topic "Modern aspects of software development" in 2019.

Publications. The abstracts were published in the collection "Modern Aspects software development" VI international scientific and practical virtual conference of software development professionals.

Keywords. AUTOMATIC CONCEPTS EXTRACTION, ONTOLOGY, CONCEPT-THESIS MODEL, AUTOMATIC TEXT PROCESSING.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Онтологія — це формалізоване представлення знань про певну предметну область (середовище, світ), придатне для автоматизованої обробки.

Концепт — це одиниця онтології (ключове слово), яка генералізує деяке поняття з навколишнього світу.

Словоформа — це складова концепту, яка являє собою одну з форм концепту (абревіатура, наприклад).

Корпус тексту — масив текстових документів.

Data mining — збір невпорядкованих даних та приведення їх у єдиний вигляд для подальшої роботи з ними.

Лематизація — це приведення слова у нормальну форму шляхом знаходження його називного відмінка.

Стемінг — це приведення слова у нормальну форму шляхом відсікання суфіксів.

ВСТУП

У сучасному світі, текстові дані займають значну частку в інформаційних системах, у тому числі у системах дистанційної освіти, де якісна структуризація знань напряму корелює з якістю всього сервісу. Одним із варіантів структуризації текстових даних являється виділення концептів та знаходження тез для них, і за допомогою такого методу можна досягнути таких цілей:

а) можливість для більш швидкої та ефективної побудови онтології;
б) побудова графового представлення між концептами для більш наочної візуалізації предметної області;
в) пошук схожих між собою документів задля кластеризації текстових даних.

Однак з ростом кількості текстових даних, ручне виділення може займати велику кількість часу, і тому постає необхідність у створенні системи для автоматичного виділення концептів та супутніх до них даних (тези, синоніми, тощо). Чинні системи для автоматичного пошуку концептів мають досить наївні алгоритми для виділення концептів, які не здатні фільтрувати неякісні варіанти, групувати їх та сортувати.

У даній роботі були розглянуті та розроблені методи для автоматичного пошуку концептів та тез до них, а також був створений REST API додаток для надання можливостей сервісу стороннім додаткам, у тому числі для веб-інтерфейсу користувача, який здатен по запиту проаналізувати вхідний текст на концепти та тези до них.

1. ПОСТАНОВКА ЗАДАЧІ

Метою розробки є створення формального апарату та програмного продукту, який автоматизує процес виділення концептів та тез на основі текстових даних на англійській мові. Також метою являється аналіз сучасних програмних продуктів та алгоритмів, які виконують поставлену задачу, зіставити їх з реаліями освітнього порталу, та, при необхідності, адаптувати зі збереженням якості їх роботи.

Робота даної системи демонструється при автоматичному зборі текстових даних з онлайн-ресурсу Semantic Portal, при автоматичному виділенні концептів та їх ранжуванню по ступеню їх важливості для контексту. Також робота демонструється шляхом знаходження відповідних тез до знайдених концептів. Для взаємодії з системою був розроблений REST API додаток, який дозволяє іншим клієнтським додаткам використовувати можливості сервісу для роботи зі своїми текстовими даними. Також для демонстрації роботи системи був розроблений веб-інтерфейс користувача, який дозволяє переглянути автоматично виділені концепти та тези до них як для будь-якої статті на Semantic Portal, так і концепти та тези для будь-якої веб-сторінки в мережі інтернет.

Розроблені алгоритми повинні якісно працювати для англомовних освітньо-орієнтованих систем, тому постає необхідність у розробці програмного рішення, яке буде враховувати специфіку статей з освітньо-орієнтованих систем. Якість роботи системи напряму залежить від кількості тексту на вході та від загальної якості його написання, тобто тексти повинні бути науково спрямовані та повинні містити значну кількість наукових термінів та їх визначень. На якість результату також впливає структуризація тексту, тобто виділення параграфів, заголовків та анотації, адже ця інформація дає велику кількість інформації для ранжування концептів.

Для виділення концептів необхідно на вхід алгоритму подати корпус тексту. Подальша обробка тексту здійснюється за допомогою наступної низки алгоритмів:

а) Токенізація та стемінг тексту.
б) Класифікація слів на частини мови для подальшого пошуку за шаблоном.
в) Фільтрація кандидатів у концепти за допомогою лінгвістичних правил та стоп-слів.

г) Видалення дублікатів серед кандидатів.
д) Ранжування кандидатів за деякою метрикою якості.
е) Пошук тез до знайдених концептів за допомогою лінгвістичних правил та сторонніх ресурсів.

Дана робота також потребує власного інтерфейсу користувача, який можна використовувати для сторонніх веб-додатків, яким необхідно виділяти концепти та тези на основі вхідних корпусів тексту. Якісна реалізація користувацького інтерфейсу потребує наявності наступних пунктів на виконання:

- вибір статей зі сторонньої бази даних;
- можливість параметризації результату роботи алгоритму (кількість концептів, флаг лематизації, інші додаткові поля з метаданими);
- створення REST API додатку для комунікації з іншими веб-сервісами;
- система авторизації у веб-додатку;
- кешування результатів виділення концептів та тез до них;
- створення веб-інтерфейсу для тестування роботи алгоритму.

Таким чином, поставлена мета потребує докладного вивчення та аналізу алгоритмів, які існують на даний момент, та їх адаптацію для англомовного освітньо-наукового порталу.

2. АНАЛІЗ ПРОБЛЕМИ АВТОМАТИЧНОГО ВИДІЛЕННЯ КОНЦЕПТІВ ТА ТЕЗ

Проблему по автоматичному виділенню концептів та тез до них являється комплексною задачею, яка має велике алгоритмічне навантаження та багато різних методів для виконання поставленої задачі.

2.1 Загальні поняття

Понятійно-тезисна модель (ПТМ) — це модель подання знань, яка формалізує зміст навчального контенту і розробляється спеціально для систем дистанційного навчання з урахуванням специфічних вимог навчального процесу [1].

Понятійно-тезисна модель розроблена для формалізації освітньо-направленого тексту і мультимедіа, що є вмістом навчальної програмної системи. ПТМ служить як засіб моделювання сенсу контенту і відповідає за формалізацію навчальних статей. ПТМ надає засоби опису понятійної складової контенту і забезпечує основу для програмного інструментарію редагування і використання бази даних та знань.

Поняття (концепт) — це певний об’єкт обговорення деякої області, який представляється для вивчення. Інакше кажучи, поняття – це одне або кілька слів, які виражають предмет розгляду деякого фрагменту навчального матеріалу [2].

Теза — це деяка інформація у вигляді набору речень про концепт. Якщо поняття вказують предмет курсу, то тези являють собою сенсове наповнення бази знань [3].

Понятійно-тезисна модель являє собою базу даних з концептами та тезами до них, між концептами можуть існувати взаємозв’язки, які можуть формувати граф, по якому можна орієнтуватися у навчальному веб-додатку. Дана структура ПТМ має велику складність, тому її наповнення призводиться вручну, що може займати значний час експерта.

Автоматичне виділення концептів та тез до них має за мету значно збільшити ефективність експерту по наповненню ПТМ у вигляді рекомендацій, які автоматично підтягуються з сервісу, який вже проаналізував статтю освітньої

направленості. Однак дані рекомендації концептів та тез до них мають деякі обмеження:

- Висока якість концептів на виході не гарантована, саме тому це має рекомендаційний характер;
- При великих обсягах тексту швидкодія роботи алгоритму може значно знизитись;
- Алгоритм призначений лише для предметної області, під яку він був створений, тому тексти з інших предметних областей можуть дати помірний результат;
- Мова тексту також має велике значення, тому передбачити роботу алгоритму для інших мов неможливо.

Однак алгоритм для автоматичного виділення концептів та тез має суттєві плюси:

- Велика швидкість обробки текстових даних;
- Повнота розбору тексту;
- Можливість підключення зовнішніх джерел даних для розширення інформації про концепт.

Таким чином, процес автоматичного виділення концептів та тез до них здатен значно пришвидшити роботу експерта по наповненню ПТМ для освітнього сервісу.

2.2 Задача автоматичного виділення концептів та тез

Автоматичне виділення концептів та тез — складний алгоритмічний процес, який повинен виконувати деякі вимоги, які можуть гарантувати якість результату та велику швидкість виконання, і тому система повинна виконувати наступні задачі:

- Паралельна обробка великих об’ємів даних (веб-ресурси, текстові файли, тощо);
- Якісна обробка тексту перед пошуком концептів для усунення непотрібних структурних елементів (HTML-теги, інші метадані);
- Токенізація тексту на речення та слова й пунктуацію;

- Виділення з тексту концептів, фільтрування та ранжування їх за мірою важливості для контексту;
- Знаходження тез для отриманих концептів;
- Кешування отриманих результатів;
- Можливість використання сторонніх сервісів для розширення інформації про концепт;

2.3 Чинні алгоритмічні рішення

Автоматичне виділення концептів та тез являється досить вузькою задачею по причині важкості її імплементації та підтримки. Однак нині існує низка алгоритмів, які виконують схожі задачі: виділення концептів, пошук тез до них, знахождення інформації про них на сторонніх ресурсах, тощо.

На даний момент, основними алгоритмами для автоматичного виділення концептів являються наступні підходи:

HUMB [4] — метод, який використовує метод навчання з вчителем. Даний алгоритм використовує **Support Vector Machines** (SVM) для класифікації слів на два класи: ”концепт“ чи ”не-концепт“. У якості входу до класифікатора виступають морфологічні параметри слів, їх положення та контекст. Плюсом даного підходу являється висока точність класифікації, а мінусом — необхідність вручну збирати велику кількість тренувальних даних для заданої предметної області.

Ентропійне ранжування концептів [5] — метод, який у своїй основі використовує поняття ентропії для ранжування концептів за їх значущостю для контексту. Плюсом даного підходу являється відсутність необхідності створювати навчальні дані для тренування алгоритму. Мінусом являється помірна якість роботи.

KPEX [6] — метод, який використовує лише евристичні метрики для ранжування концептів, такі як довжина слова, його частота в тексті, абревіатура чи ні. Плюсом даного підходу являється велика гнучкість у параметрах алгоритму, можна додавати ті чи інші метрики. Мінусом являється його плюс, адже для його адаптації для предметної області необхідно вручну підібрати його параметри для його якісної роботи.

Всі вищеназвані алгоритми можна розширити на будь-яку мову, однак використовуються вони для аналізу наукових статей, тобто методів для виключно освітньо-направлених текстів на даний не існує.

2.4 Огляд алгоритмів синтаксичного аналізу тексту для виділення цільових сутностей

Задача по автоматичному виділенню концептів потребує розробки специфічних алгоритмів для роботи з текстовими даними, починаючи з токенізації й закінчуючи методом фільтрації виключно важливих концептів з тексту та пошуком тез до них.

Метод автоматичного виділення концептів являє собою послідовність з низки алгоритмів, які поступово перетворюють текст у компактну структуру даних, яка зберігає знайдені концепти та тези до них. Загальні етапи по обробці текстового документа виглядають для всіх мов та для всіх предметних областей наступним чином:

а) токенізація (розбиття тексту на речення, слова й пунктуацію);
б) класифікація слів на частини мови (необхідна для подальшого виділення концептів за заданим шаблоном);
в) виділення кандидатів у концепти за заданим шаблоном (регулярний вираз);
г) фільтрація концептів за стоп-словами чи шаблонами;
д) ранжування концептів за їх важливістю для контексту та їх нормалізація;
е) пошук тез до концептів.

Таким чином, алгоритм для автоматичного виділення концептів та тез являється комплексом з багатьох інших алгоритмів, які доповнюють один одного та використовуються на різних етапах обробки вхідного тексту.

2.4.1 Токенізація тексту

Токенізація тексту — це один з найпростіших етапів обробки електронного тексту, це алгоритм ділить текст на слова та пунктуацію.

Електронний текст — це лінійна послідовність символів. Перед тим, як здійснити будь-яку реальну обробку тексту, текст потрібно сегментувати на мовні одиниці, такі як слова, пунктуація, цифри, інші символи тощо. Цей процес називається токенізацією.

Ідентифікація одиниць, які не потребують подальшого розкладання для подальшої обробки, є надзвичайно важливою. Помилки, допущені на цьому етапі, швидше за все можуть викликати більше помилок на пізніших етапах обробки тексту, і тому є дуже небезпечними.

Існує багато видів токенізації, і найпростішим варіантом являється розділення по пробілах та видалення всієї пунктуації. Цей підхід використовує full text search рушій **Lucene** [7], адже цей метод здатен оброблювати значні по своїм об'ємам тексти за секунди реального часу. Цей підхід підходить для швидкого пошуку документів серед корпусу документів, однак для задачі по виділенню концептів цей метод видаляє велику кількість корисної інформації.

Токенізація, як правило, вважається легкою щодо інших завдань обробки мов, і одним із найважніших задач (для англійської та інших сегментованих мов). Однак помилки, допущені на цій фазі, можуть призвести до помилок у наступних фазах. Для вирішення цієї проблеми було розроблено множину вдосконалених методів, які вирішують конкретні проблеми токенізації для доповнення стандартних токенізаторів.

Боб Карпентер зазначає, що токенізація особливо неприємна в області біомедичного тексту, де є багато слів (або принаймні фразових лексичних записів), які містять дужки, дефіси тощо, тобто являють собою складні хімічні формули.

Іншим викликом для токенізації є "брудний текст". Не весь текст був перевірений у процесі редагування та перевірки орфографії. Текст, вилучений автоматично з PDF-файлів, полів баз даних чи інших джерел, може містити неточно складені лексеми, орфографічні помилки та несподівані символи. У деяких випадках, коли текст зберігається в базі даних у фіксованих полях з декількома рядками на об'єкт, поля іноді потрібно повторно зібрати, однак пробіли необхідно обробити.

Першим кроком у більшості програм для обробки тексту є сегментація тексту на слова.

У всіх сучасних мовах, які використовують латинську, кириличну або грецьку систему письма, наприклад, англійська та інші європейські мови, лексеми слів відрізняються наявністю пробіла. Таким чином, для таких мов, які називаються сегментованими мовами, ідентифікація границь лексеми є дещо тривіальною завданням, оскільки більшість лексем пов'язані явними роздільниками, як пробіли та пунктуація. Проста програма, яка замінює пробіли межею слів

і обрізає провідні та кінцеві лапки, круглі дужки та пунктуації, вже дає високу ефективність.

Більшість існуючих токенізаторів сигналізують межі лексеми пробілами. Таким чином, якщо такий токенізатор знаходить два лексеми, що знаходяться безпосередньо поруч один з одним, як, наприклад, коли за словом йде кома, він вставляє між ними пробіл.

Концептуально розщеплення на пробіли також може розділити те, що слід розглядати як єдиний маркер. Найчастіше це трапляється з іменами (Сан-Франциско, Лос-Анджелес), але також із запозиченими іноземними фразами (au fait) та сполуками, які іноді пишуться як одне слово, а іноді розділяються пробілом (наприклад, пробіл проти пробілу). Інші випадки із внутрішніми пробілами, які ми могли б розглянути як єдиний маркер, включають номери телефонів ((555) 234-2333) та дати (11 березня 1983 р.). Розбиття токенів на пробіли може спричинити погані результати пошуку, наприклад, якщо пошук Йоркського університету в основному повертає документи, що містять Нью-Йоркський університет. Оголошення про тарифи на авіаквитки часто містять такі предмети, як Сан-Франциско-Лос-Анджелес, де просто розбиття по пробілах дало б погані результати. У таких випадках задачі токенізації взаємодіють з обробкою фразових запитів, особливо якщо ми хочемо, щоб запити для всіх малих регістрів давали однакові результати. Останні два можна обробити розділенням дефісів та використанням індексу фрази. Отримання правильного першого випадку залежатиме від того, щоб знати, що його іноді записують як два слова, а також індексувати таким чином. Одна ефективна стратегія на практиці, яка використовується деякими булевими системами пошуку, такими як Westlaw та Lexis-Nexis (westlaw), — це заохочувати користувачів вводити дефіси, де це можливо, і всякий раз, коли є дефіс, система буде узагальнюватися. запит, який охоплює всі три слова, дефіс та дві словоформи, так що запит на надмірного нетерпіння буде шукати надмірного чи АБО ”над охочим” АБО надмірного. Однак ця стратегія залежить від навчання користувачів, оскільки якщо ви запитуєте за допомогою будь-якої з двох інших форм, ви не отримаєте узагальнення.

Припущення щодо того, що вихідний текст буде ідеальним, може мати погані наслідки. Токенізатор часто повинен бути налаштований під ці дані.

Визначення того, чи повинні два чи більше слів стояти разом, щоб утворити один маркер (наприклад, “Раціональний архітектор програмного забезпечення”),

було би завданням токенізації високого рівня. Сегментація високого рівня набагато більш мовно мотивована, ніж сегментація “низького рівня”, і вимагає (як мінімум) відносно дрібної мовної обробки.

Інший варіант алгоритму токенізації може враховувати пунктуацію між словами та пунктуацію в самих словах (дефіси, апострофи). Такий алгоритм використовується для більш детального розбору тексту, який на наступних этапах буде використовуватись для різних цілей: виділення концептів, знаходження іменованих сутностей, тощо.

Всі вищеназвані алгоритми для токенізації тексту можуть використовувати як мову регулярних виразів, так і підхід, який заснований на ієрархічному розділенні слова, тобто спочатку текст ділится на слова за пробілами, далі отримані слова діляться за базовою пунктуацією, далі за дефісом і т.д., тобто є можливість контролювати процес та зупинити токенізацію на будь-якому рівні.

Ідентифікація одиниць, які не потребують подальшого розкладання для подальшої обробки, є надзвичайно важливою. Помилки, допущені на цьому етапі, швидше за все можуть викликати більше помилок на пізніших етапах обробки тексту, і тому є дуже небажаними.

В залежності від цілей, токенізація може бути “грубою” (ділити слова з дефісом та апострофом на токени) або “м’якою” (зберігати слова с дефісом чи апострофом). Найпростішим токенізатором являється саме просте розбиття слів по пробілах. Однак ця реалізація не підходить для “грубої” токенізації, адже вона не зможе розбити слова по деякій внутрішній пунктуації (дефіси). Для задачі автоматичного виділення концептів та тез найкраще підходить “м’яка” токенізація, адже у подальшій роботі стає необхідність перетворювати токени ключового слова у строку зі збереженням форматування.

2.4.2 Класифікація слів по частинах мови

Класифікація слів за частинами мови (part of speech tagging) — це присвоєння кожному слову у тексті відповідну частину мови. Ця задача являється досить комплексною, адже деякі слова у різних контекстах можуть бути різними частинами мови. Тому для класифікації також необхідно враховувати контекст, який оточує слово. Контекст — це декілька слів справа та зліва.

Для безпосередньо класифікації використовують методи машинного навчання, тобто створюють навчальну спеціальну вибірку, де у якості вхідних даних йде інформація про слово (чи великими літерами написане, чи присутні дефіси чи

числа, чи входить це слово в список пунктуаційних символів, тощо) та про його контекст (аналогічні параметри, які для власне самого слова). Всі ці параметри кодують у векторні числові представлення, які вже згодом йдуть в алгоритм машинного навчання.

При обробці мов кожне слово в реченні позначається своєю частиною мови. Потім ці теги стають корисними для додатків вищого рівня. Поширеними частинами мови в англійській мові є іменник, дієслово, прикметник, прислівник тощо.

Основна проблема з тегами частин мови — це неоднозначність. В англійській мові багато загальних слів мають багато значень і, отже, декілька частин мови. Завдання маркера частини мови полягає у вирішенні цієї неоднозначності точно, виходячи з контексту використання слова. Наприклад, слово “shoot” може бути іменником або дієсловом. Якщо воно використовується як дієслово, воно може бути в минулому часі чи дієприкметнику минулого часу.

Маркери частин мови взяли свій початок з лінгвістичного підходу, але пізніше перейшли до статистичного підходу. Сучасні моделі досягають точності більше, ніж 97%. Дослідження маркерів частин мови, проведені з англійським текстовим корпусом, було адаптовано до багатьох інших мов.

Класифікатор на частини мови містить фразу або речення і присвоює кожному кожному слові відповідний тег. На практиці, вхідний текст часто попередньо обробляється. Одним із загальних завдань перед обробкою є токенізація тексту, щоб класифікатор бачив послідовність слів і пунктуації. Перед тегом можуть бути виконані інші задачі, такі як видалення стоп-слів, видалення пунктуації та лематизація.

Набір попередньо визначених тегів називається набором тегів. Це важлива інформація про те, на які класи буде класифікувати класифікатор на частини мови. Приклад тегів — NNS для іменника множини, VBD для дієслова минулого часу або JJ для прикметника. Набір тегів також може містити пунктуацію.

Замість того, щоб розробити власний набір тегів, загальною практикою є використання відомих наборів тегів: 87-тегів з Brown корпусу , 45-тегів з Penn Treebank, 61-тегів з C5-набору або 146-тегів з C7-набору.

Класифікація на частини мови — це основне завдання в NLP. Це важливе завдання попередньої обробки перед тим, як робити синтаксичний аналіз або семантичний аналіз. Він приносить користь багатьом програмам NLP, включаючи

пошук інформації, вилучення інформації, системи мовлення тексту, мовознавство корпусу, розпізнавання названих об'єктів, відповіді на запитання, розбір слова в сенсі і багато іншого.

Якщо тег POS дає низьку точність, це негативно впливає на інші завдання, які випливають. Це зазвичай називають розповсюдженням помилок за течією. Для підвищення точності деякі дослідники запропонували поєднувати позначення POS з іншою обробкою. Наприклад, спільне тегування POS та розбір залежності — це підхід до підвищення точності порівняно з незалежним моделюванням.

Іноді слово самостійно може дати корисні підказки. Наприклад, "the" є визначальним фактором. Префікс "un-" передбачає прикметник, наприклад, "unathomable". Суфікс "-ly" пропонує прислівник, наприклад "importantly". Пропис з великої літери може підказати власне іменник, наприклад, "меридіан". Корисні також словоформи, наприклад, "35 -річний", який є прикметником.

Слово можна позначати на основі сусідніх слів та можливих тегів, які ці слова можуть мати. Вірогідності слова також грають важливу роль у виборі правильного тегу для усунення неоднозначності. Наприклад, 'людина' рідко використовується як дієслово і в основному використовується як іменник.

При статистичному підході ми можемо порахувати частоту тегів слів у позначеному корпусі, а потім призначити найбільш ймовірний тег. Це називається тегом уніграма. Набагато кращим підходом є позначення біграми. При цьому враховується частота тегів, що задається певним попереднім тегом. Таким чином, тег, як видно, має залежність від попереднього тегу. Ми можемо узагальнити це тегами n-грамів. Насправді, звичайно моделювати послідовність слів і оцінювати послідовність тегів. Це робиться за допомогою **Hidden Markov Model** (HMM) [8].

Існують наступні види адгоритиів для класифікації на частини мови:

- **rule-Based**: Словник будується з можливими тегами для кожного слова. Правила керують тегером для розмежування. Правила є ручними, вивченими або обома. Приклад правила може сказати: "Якщо двозначному / невідомому слову X передує визначник, а за ним іменник, позначте його як прикметник";
- **statistical**: Текстовий корпус використовується для отримання корисних ймовірностей. Враховуючи послідовність слів, вибирається найбільш вірогідна послідовність тегів. Вони також називаються стохастичними або ймо-

вірнісними мітками. Серед поширених моделей — n-грамова модель, прихована модель Маркова (HMM) та модель максимальної ентропії (MEM);

- **memory-Based**: набір справ зберігається в пам'яті, кожен випадок містить слово, його контекст і відповідний тег. Нове речення позначено тегами на основі найкращої відповідності випадкам, збереженим у пам'яті. Це поєднання методологічного та стохастичного методів;
- **transformation-Based**: правила автоматично індукуються з даних. Таким чином, це поєднання методів, заснованих на правилах і стохастичних. Позначення тегів проводиться за допомогою широких правил, а потім покращується або трансформується, застосовуючи більш вузькі правила;
- **Neural network**: RNN та двонаправлений LSTM — два приклади архітектури нейронної мережі для мітки POS.

Класифікатори на частини мови можуть бути як під наглядом, так і без нагляду. Контрольовані теги покладаються на тег корпусу для створення словника, правил або ймовірностей послідовності тегів. Вони найкраще працюють, коли навчаються та застосовуються в одному жанрі тексту. Непідконтрольні мітки індукують групування слів. Це економить зусилля попереднього позначення корпусу, але скупчення слів часто грубі.

Поєднання обох підходів також є загальним. Наприклад, правила автоматично індукуються з немеченого корпусу. Вихід з цього виправляється людиною і повторно подається на теггер. Теггер переглядає виправлення та коригує правила. Багато ітерацій цього процесу можуть бути необхідними.

У 2016 році було відмічено, що повністю непідконтрольний підхід ще не зрілий. Натомість слабко контрольовані підходи приймаються шляхом вирівнювання тексту, використання ймовірностей перекладу (для машинного перекладу) або передачі знань з мов, багатих ресурсами. Навіть невелику кількість міченого корпусу можна узагальнити, щоб дати кращі результати.

Це називається це проблемою декодування. Ми можемо спостерігати послідовність слів, але послідовність тегів прихована. Нам потрібно з'ясувати найбільш ймовірну послідовність тегів, задану послідовністю слів. Іншими словами, ми хочемо максимізувати $P(t^n|w^n)$ для послідовності n слів.

Важливим розумінням є те, що частини мови (а не слова) надають мові її структуру. Таким чином, використовуючи правило Байєса, ми переробили задачу

на наступну форму: $P(t^n|w^n) = P(w^n|t^n)P(t^n)/P(w^n)$. $P(w^n|t^n)$ називається ймовірністю. $P(t^n)$ називається апріорною ймовірністю.

Оскільки ми максимізуємо всі послідовності тегів, знаменник можна ігнорувати. Ми також робимо два припущення: кожне слово залежить лише від власного тегу, а кожен тег залежить лише від його попереднього тегу. Тому нам потрібно максимізувати $\prod_{n=1}^{n} P(w_i^n|t_i^n)P(t_i|t_{i-1})$. У HMM терміни називаються ймовірностями викидів та ймовірностями переходу.

Ці ймовірності оцінюються з тегів текстового корпусу. Стандартним рішенням є застосування алгоритму Вітербі, який є формою динамічного програмування. На прикладі малюнка ми бачимо два ненульові шляхи і вибираємо більш ймовірний.

У будь-якому контрольованому статистичному підході рекомендується розділити свій корпус на навчальний набір, набір розробок (для настройки параметрів) та набір для тестування. Альтернативою є використання всього корпусу для навчання, але робити перехресну перевірку. Крім того, якщо корпус занадто загальний, ймовірності можуть не відповідати певній галузі; якщо він занадто вузький, він може не узагальнити між доменами. Щоб проаналізувати, де ваша модель виходить з ладу, ви можете використовувати матрицю плутанини або таблицю надзвичайних ситуацій.

Коли бачать невідомі слова, одним із підходів є призначення суфіксу та обчислення ймовірності того, що суфікс-слово з певним тегом відбувається в послідовності. Інший підхід полягає у призначенні набору тегів за замовчуванням та обчисленні ймовірностей. Або ми можемо переглянути внутрішню структуру слова, наприклад, призначити NNS для слів, що закінчуються на “s”.

Для боротьби з розрідженими даними (ймовірності дорівнюють нулю) існують методи згладжування. Наївна техніка — додати невелику кількість частот, скажімо, 1, до всіх показників. Метод Доброго Тюрінга разом із баккофтом Каца є кращою методикою. Лінійна інтерполяція — ще одна методика.

Англійською мовою, що вивчає, є англійська мова як іноземна. Такий текст часто містить орфографічні та орфографічні помилки. Використання нейронних мереж, зокрема двонаправленого LSTM, забезпечує кращу точність, ніж стандартні теги POS. Використовуються вбудовані слова, вставки символів та вектори рідної мови.

Історична англійська також представляє складні проблеми через різницю в написанні, вживанні та лексиці. Поєднання нормалізації написання та методу адаптації домену, такого як вбудовування функцій, дає кращі результати. Інші підходи до історичного тексту включають нейронні сітки, умовні випадкові поля та методи самонавчання.

Винайдено методи для позначення даних Twitter, які часто рідкісні та галасливі. У математиці POS-теги були адаптовані для обробки формул та вилучення ключових фраз у математичних публікаціях. Для клінічного тексту використовується маркований корпус для цього жанру. Однак було встановлено, що краще ділитися анотаціями між корпораціями, ніж просто ділитися попередньо перевіреною моделлю.

У Python пакет nltk.tag реалізує багато типів тегів. Шаблон - це модуль вебмайнінгу, який включає можливість робити теги POS. На жаль, йому не вистачає підтримки Python 3. Він також доступний в R як pattern.nlp. TextBlob надихається як NLTK, так і шаблоном. spaCy — ще один корисний пакет.

Реалізований у TensorFlow, SyntaxNet базується на нейронних мережах. Parsey McParseface — це аналізатор англійської мови і дає хорошу точність.

У школах зазвичай навчають, що в англійській мові є 9 частин мови: іменник, дієслово, стаття, прикметник, прийменник, займенник, прислівник, сполучник та вилучення. Однак, очевидно, є ще багато категорій та підкатегорій. Для іменників можна виділити множинні, присвійні та однинні форми. У багатьох мовах слова також позначаються як їх ”відмінок” (роль як предмет, предмет тощо), граматична стать тощо; в той час як дієслова позначаються за часом, аспектом та іншими речами. У деяких системах тегування різні переклади одного і того ж кореневого слова отримають різні частини мови, в результаті чого з’явиться велика кількість тегів. Наприклад, NN для спільних іменників однини, NNS для множинних загальних іменників, NP для однини власних іменників однини (див. POS-теги, які використовуються у Коричневому корпусі). Інші системи тегування використовують меншу кількість тегів і ігнорують тонкі відмінності або моделюють їх як функції, дещо незалежні від часткової мови.

У розділі тегів мовлення за комп’ютером типово розрізняти від 50 до 150 окремих частин мови для англійської мови. Робота над стохастичними методами для позначення Койн Грека використовувала понад 1000 частин мови і виявила, що приблизно стільки слів були неоднозначними в цій мові, як і англійською.

Морфосинтаксичний дескриптор у випадку морфологічно багатих мов зазвичай виражається за допомогою дуже коротких мнемоніків, таких як Ncmsan для категорії = іменник, тип = загальний, стать = чоловічий, число = однини, випадок = звинувачувальний, анімаційний = ні.

Найпопулярніший "набір тегів" для тегів на POS для американської англійської мови - це, мабуть, набір тегів Пенн, розроблений у проекті Penn Treebank. Він багато в чому схожий з попередніми наборами тегів Brown Corpus та LOB Corpus, хоча і значно менший. У Європі набори тегів із Посібників Eagles широко використовуються та містять версії для декількох мов.

Робота над тегами POS проводиться на різних мовах, а набір використовуваних тегів POS значно відрізняється від мови. Теги, як правило, розроблені так, щоб включати явні морфологічні відмінності, хоча це призводить до невідповідностей, таких як маркування букв для займенників, але не іменників в англійській мові, та набагато більших міжмовних відмінностей. Набори тегів для сильно перетворених мов, таких як грецька та латинська, можуть бути дуже великими; позначення слів на агглютинативних мовах, таких як інуїтські мови, може бути практично неможливим. З іншого боку, Петров та ін. запропонували "універсальний" набір тегів з 12 категоріями (наприклад, відсутні підтипи іменників, дієслів, пунктуації тощо; відсутність відміни "до" як інфінітивний маркер проти прийменника (навряд чи "універсальний" збіг), тощо). Переважний, чи дуже невеликий набір дуже широких тегів або набагато більший набір більш точних, залежить від конкретної мети. На менших наборах тегів автоматичне тегування простіше.

Дослідження тегів, що належать до мови, були тісно пов'язані з корпусним мовознавством. Першим великим корпусом англійської мови для комп'ютерного аналізу був коричневий корпус, розроблений в університеті Браун Генрі Кучерою та У. Нельсоном Френсісом в середині 1960-х. Він складається з близько 1 000 000 слів англійського прозового тексту, складених з 500 зразків з випадково вибраних публікацій. Кожен зразок - 2000 і більше слів (закінчується в першому реченні після кінця 2000 слів, так що корпус містить лише повні речення).

Протягом багатьох років Коричневий Корпус ретельно "позначався" маркерами часткової мови. Перше наближення було зроблено програмою Гріна та Рубіна, яка складалася з величезного переліку ручної роботи про те, які категорії взагалі можуть існувати. Наприклад, може виникати іменник article, але

дієслово Article (можливо) не може. Програма отримала близько 70% коректності. Її результати неодноразово переглядалися та виправлялися вручну, а пізніше користувачі надсилали помилки, так що до кінця 70-х маркування було майже ідеальним (допускаючи деякі випадки, коли навіть люди, які виступають не можуть погодитися).

Цей корпус був використаний для безлічі досліджень частоти слів і часткової мови і надихнув на розвиток подібних "мічених" корпусів у багатьох інших мовах. Статистика, отримана шляхом аналізу, лягла в основу більшості пізніших систем тегування мовлення, що належить до мови, таких як CLAWS (лінгвістика) та VOLSUNGA. Однак до цього часу (2005 р.) Його витіснили більші корпорації, такі як Британський національний корпус на 100 мільйонів слів.

Деякий час маркування частиною мови вважалося невіддільною частиною природного опрацювання мови, оскільки є певні випадки, коли правильну частину мови неможливо визначити без розуміння семантики чи навіть прагматики контексту. Це надзвичайно дорого, тим більше, що аналізувати вищі рівні набагато складніше, коли для кожного слова необхідно враховувати кілька можливостей, що належать до мови.

З середини 1980-х дослідники Європи почали використовувати приховані марківські моделі (HMM) для розмежування частин мови, працюючи над позначенням англійського англійського корпусу Ланкастера-Осло-Берген. HMM включають підрахунок випадків (наприклад, з коричневого корпусу) та складання таблиці ймовірностей певних послідовностей. Наприклад, щойно ви побачили таку статтю, як "the", можливо, наступне слово — це іменник 40% часу, прикметник 40% і число 20%. Знаючи це, програма може вирішити, що "can" у "can" набагато частіше є іменником, ніж дієсловом чи модалом. Цей же метод, звичайно, може бути використаний для отримання знань про наступні слова.

Більш просунуті ("вищого порядку") HMM вивчають ймовірності не лише пар, але й потрійних чи навіть більших послідовностей. Так, наприклад, якщо ви щойно бачили іменник, за яким слідує дієслово, наступним пунктом може бути, швидше за все, прийменник, стаття чи іменник, але набагато рідше інший дієслово.

Коли кілька неоднозначних слів трапляються разом, можливості множиться. Однак легко перерахувати кожну комбінацію і призначити відносну ймовірність кожній з них, множивши по черзі ймовірності кожного вибору. Потім вибирається

комбінація з найбільшою ймовірністю. Європейська група розробила CLAWS - програму мічення, яка саме це і зробила, і досягла точності в діапазоні 93–95%.

Варто пам'ятати, як Євген Чарняк зазначає у статистичних методах розбору природних мов, що лише присвоєння найвідомішого тегу кожному відомому слову, а тег "власний іменник" усім невідомим наблизиться до 90% точності оскільки багато слів однозначні, а багато інших лише рідко представляють їх менш поширені частини мови.

CLAWS першопрохідцем в області мовлення, що базується на HMM, але була досить дорогою, оскільки вона перераховувала всі можливості. Іноді доводилося вдаватися до методів резервного копіювання, коли було просто занадто багато варіантів (Коричневий корпус містить випадок із 17 неоднозначними словами підряд, а є такі слова, як "все ще", які можуть представляти цілих 7 різних частин мови.

У 1987 році Стівен Дероз та Кен Черч незалежно розробили алгоритми динамічного програмування для вирішення однієї і тієї ж проблеми за значно менший час. Їх методи були схожі на алгоритм Вітербі, відомий певний час в інших сферах. DeRose використовував таблицю пар, в той час як Черч використовував таблицю потрій і метод оцінки значень для трійки, які були рідкісні або відсутні в Корпусі Брауна (фактичне вимірювання потрійних ймовірностей вимагало б набагато більшого корпусу). Обидва методи досягли точності понад 95%. Дисертація Дероза 1990 року в Університеті Брауна включала аналіз конкретних типів помилок, ймовірностей та інших пов'язаних даних і повторила його роботу на грецькій мові, де вона виявилася аналогічно ефективною.

Ці результати були напрочуд руйнівними для галузі обробки природних мов. Точність, про яку повідомлялося, була вищою, ніж типова точність дуже складних алгоритмів, які інтегрували частину вибору мови з багатьма вищими рівнями мовного аналізу: синтаксисом, морфологією, семантикою тощо. Методи КЛАВС, Дероза та Церкви зазнали невдачі для деяких відомих випадків, коли потрібна семантика, але вони виявились незначно рідкісними. Це переконувало багатьох у цій галузі в тому, що теги на частину мови можна корисно відокремити від інших рівнів обробки; це, у свою чергу, спростило теорію та практику комп'ютеризованого аналізу мови та спонукало дослідників знайти способи відокремити й інші фрагменти. Моделі Маркова тепер є стандартним методом для часткового мовлення.

Методи, які вже обговорювались, передбачають роботу з уже існуючого корпусу для вивчення ймовірностей тегів. Однак можливо також завантажувати завантаження за допомогою “непідконтрольного” тегування. Непідконтрольні методи тегування використовують немечений корпус для своїх навчальних даних та виробляють набір тегів шляхом індукції. Тобто вони дотримуються закономірностей у вживанні слова та самі виводять частки мови. Наприклад, статистика легко показує, що “а” і “і” трапляються в подібних контекстах, тоді як “їдять” трапляються в дуже різних. При достатній ітерації виходять класи подібності слів, які надзвичайно схожі на очікування людських лінгвістів; а самі розбіжності інколи підказують цінні нові знання.

Ці дві категорії можна додатково підрозділити на основі правил, стохастичних та нейронних підходів.

Деякі основні поточні алгоритми тегування часткової мови включають алгоритм Вітербі, тест Бриля, Граматика обмежень та алгоритм Баума-Велча (також відомий як алгоритм прямого назад). Прихована модель Маркова та видимі маркерські моделі Маркова можуть бути реалізовані за допомогою алгоритму Вітербі. Теглер Brill на основі правил незвичний тим, що він вивчає набір шаблонів правил, а потім застосовує ці шаблони, а не оптимізує статистичну величину. На відміну від тегеля Brill, де правила впорядковуються послідовно, POS та морфологічний інструментарій тегів RDRPOSTagger зберігає правила у вигляді дерева правил пульсації.

До проблеми POS-тегування також застосовано багато методів машинного навчання. Такі методи, як SVM, максимальний класифікатор ентропії, перцептрон і найближчий сусід, були випробувані, і більшість може досягти точності вище 95%.

Пряме порівняння декількох методів повідомляється (з посиланнями) на ACL Wiki. У цьому порівнянні використовується тег Penn, встановлений на деяких даних Penn Treebank, тому результати прямо порівнянні. Однак багато значущих тегів не включаються (можливо, через працю, пов’язану з перенастроюванням їх для цього конкретного набору даних). Таким чином, не слід вважати, що результати, про які повідомляється, є найкращими, що можна досягти за допомогою даного підходу; ні навіть найкращого, що було досягнуто заданим підходом.

У 2014 році звіт про папір, що використовує метод регуляризації структури для тегування частини мови, досягаючи 97,36% у стандартному наборі даних орієнтиру.

CRF — це дискримінаційні імовірнісні класифікатори. Різниця між дискримінаційною та генеративною моделями полягає в тому, що хоча дискримінаційні моделі намагаються моделювати умовний розподіл ймовірностей, тобто $P(y|x)$, генеративні моделі намагаються моделювати спільний розподіл ймовірностей, тобто $P(x, y)$.

Логістична регресія, SVM, CRF — це дискримінаційні класифікатори. Наївні Байєси, HMM — це генеративні класифікатори. CRF також можна використовувати для завдань маркування послідовностей, таких як Named Entity Recognisers та POS Taggers.

У CRF, вхід — це сукупність ознак (реальних чисел), отриманих із послідовності введення з використанням функціональних функцій, ваг, пов'язаних з ознаками (які вивчаються) та попередньої мітки, і завдання полягає в тому, щоб передбачити поточну мітку. Ваги різних функціональних функцій визначатимуться таким чином, що ймовірність міток у навчальних даних буде максимальною.

У CRF визначений набір функціональних функцій для отримання функцій для кожного слова в реченні. Деякі приклади функціональних функцій: це перша літера слова з великої літери, що таке суфікс і префікс слова, яке попереднє слово, це перше чи останнє слово речення, чи є це число тощо. Набір функцій називається "Особливості держави". У CRF ми також передаємо мітку попереднього слова та мітку поточного слова для вивчення ваг. CRF намагатиметься визначити ваги різних функціональних функцій, які дозволять максимально збільшити ймовірність міток у навчальних даних. Функція функції, що залежить від мітки попереднього слова, - Transition Feature.

МГМ лежать в основі функціонування стохастичних міток і використовуються в різних алгоритмах, одним з найбільш широко використовуваних є алгоритм двонаправленого висновку.

Для безпосередньо класифікації використовують методи машинного навчання, тобто створюють навчальну спеціальну вибірку, де у якості вхідних даних йде інформація про слово (чи великими літерами написане, чи присутні дефіси чи числа, чи входить це слово в список пунктуаційних символів, тощо) та про його контекст (аналогічні параметри, які для власне самого слова). Всі ці параметри

кодують у векторні числові представлення, які вже згодом йдуть в алгоритм машинного навчання.

2.4.3 Виділення цільових сутностей по шаблонах

Пошук ключових слів — найважливіша частина автоматичного виділення концептів та тез до них, адже саме від нього залежить якість кінцевого результату. Однак структура натуральних мов така, що тут необхідно використовувати регулярні вирази для пошуку заданих паттернів. Звичайні регулярні вирази по буквах у цьому випадку не можуть знадобитись, адже необхідно також враховувати порядок частин мов. Тому постає необхідність по створенню власного механізму по пошуку необхідної послідовності слів.

Регулярні вирази виникли в 1951 році, коли математик Стівен Коул Клійн описав звичайні мови, використовуючи свої математичні позначення під назвою регулярні події. Вони виникли в теоретичній інформатиці, в підполях теорії автоматів (моделі обчислення) та описі та класифікації формальних мов. Інші ранні впровадження відповідності шаблонів включають мову SNOBOL, який не використовував регулярні вирази, а натомість власні конструкції відповідності шаблону.

Регулярні вирази увійшли до популярного використання з 1968 р. У двох напрямках: узгодження зразків у текстовому редакторі та лексичний аналіз у компіляторі. [7] Серед перших появи регулярних виразів у програмній формі було, коли Кен Томпсон вбудував позначення Клейна в редактор QED як засіб узгодження шаблонів у текстових файлах. Для швидкості Томпсон здійснив регулярне зіставлення виразів за допомогою щомісячної компіляції (JIT) з кодом IBM 7094 в сумісній системі обміну часом, важливим раннім прикладом компіляції JIT. Пізніше він доповнив цю можливість редактором Unix ed, що врешті-решт призвело до використання звичайних інструментів пошуку grep для регулярних виразів (“grep” — слово, отримане з команди для пошуку регулярних виразів у редакторі ed: g / re / p значення) “Глобальний пошук для відповідних рядків регулярного вираження та друку”). Приблизно в той же час, коли Томпсон розробив QED, група дослідників, включаючи Дугласа Т. Росса, реалізувала інструмент, заснований на регулярних виразах, який використовується для лексичного аналізу в дизайні компілятора.

Багато варіантів цих оригінальних форм регулярних виразів були використані в програмах Unix в Bell Labs у 1970-х роках, включаючи vi, lex, sed, AWK та

expr, та в інших програмах, таких як Emacs. Згодом Regexes були прийняті широким спектром програм, причому ці ранні форми були стандартизовані стандартом POSIX.2 у 1992 році.

У 1980-х рр. У Perl виникли більш складні регекси, які спочатку походять з бібліотеки регексів, написаної Генрі Спенсером (1986), який пізніше написав реалізацію розширених регулярних виразів для Tcl. Бібліотека Tcl — це гібридна реалізація NFA / DFA з покращеними характеристиками продуктивності. Програмні проекти, які застосували регулярну реалізацію Tcl Spencer Tcl, включають PostgreSQL. Пізніше Perl розширив оригінальну бібліотеку Спенсера, щоб додати багато нових функцій. Частина зусиль у розробці Raku полягає у вдосконаленні інтеграції регулярних виразів Perl, а також у збільшенні їх обсягу та можливостей, що дозволяють визначити граматики вираження синтаксичного аналізу. Результатом є міні-мова під назвою правила Раку, яка використовується для визначення граматики Раку, а також надання інструменту програмістам на мові. Ці правила підтримують існуючі особливості виразів Perl 5.x, але також дозволяють визначити BNF-стилі рекурсивного аналізатора спуску за допомогою підправил.

Використання регексів у структурованих інформаційних стандартах для моделювання документів і баз даних розпочалося в 1960-х роках і розширилося в 1980-х роках, коли консолідовані такі галузеві стандарти, як ISO SGML (попередній ANSI “GCA 101-1983”). Ядро стандартів мови специфікації структури складається з регулярних виразів. Його використання очевидно в синтаксисі групи елементів DTD.

Починаючи з 1997 року, Філіп Хейзл розробив PCRE (Perl Compatible Regular вирази), який намагається тісно імітувати функцію регулярних виразів Perl і використовується багатьма сучасними інструментами, включаючи PHP і Apache HTTP Server.

Сьогодні широко використовуються регулярні вирази в мовах програмування, програмах обробки тексту (зокрема лексемах), вдосконалених текстових редакторах та деяких інших програмах. Підтримка Regex є частиною стандартної бібліотеки багатьох мов програмування, включаючи Java та Python, і вбудована в синтаксис інших, включаючи Perl та ECMAScript. Реалізація функціональних можливостей регулярного генезу часто називається двигуном регулярного вирівнювання, і ряд бібліотек доступні для повторного використання. В кінці 2010-х

декілька компаній почали пропонувати апаратні, FPGA, GPU реалізації сумісних з PCRE двигунів-регексів, які швидше порівняно з реалізацією процесора.

В інформатиці алгоритм побудови Томпсона, який також називають алгоритмом Мак-Нофтона-Ямада-Томпсона, є методом перетворення регулярного виразу в еквівалентний недетермінований кінцевий автомат (NFA). Цей NFA може використовуватися для зіставлення рядків із регулярним виразом. Цей алгоритм зараховується до Кена Томпсона.

Регулярні вирази та недетерміновані кінцеві автомати є двома представленнями формальних мов. Наприклад, утиліти для обробки тексту використовують регулярні вирази для опису розширених моделей пошуку, але NFA краще підходити для виконання на комп'ютері. Отже, цей алгоритм представляє практичний інтерес, оскільки він може складати регулярні вирази в NFA. З теоретичної точки зору, цей алгоритм є частиною доказу того, що вони обидва приймають абсолютно однакові мови, тобто звичайні мови.

NFA може бути детермінований конструкцією енергетичного набору, а потім мінімізуватися для отримання оптимального автомата, відповідного даному регулярному виразу. Однак НФА може також тлумачитися безпосередньо.

Щоб вирішити, чи описані два задані регулярні вирази однією і тією ж мовою, кожен може бути перетворений в еквівалентний мінімальний детермінований кінцевий автомат за допомогою побудови Томпсона, побудови потужності та мінімізації DFA. Якщо і лише якщо отримані автомати згодні на перейменування штатів, мови регулярних виразів погоджуються.

Алгоритм працює рекурсивно, розбиваючи вираз на його складові субвирази, з яких буде побудовано NFA за допомогою набору правил. Точніше, з регулярного виразу E отриманий автомат A з функцією переходу δ дотримується таких властивостей:

- A має рівно один початковий стан q_0, який недоступний для будь-якого іншого стану. Тобто, для будь-якого стану q та будь-якої літери a, δ (q, a) $\delta(q, a)$ не містить q_0;
- A має рівно один кінцевий стан q_f, який не є спільним для доступу до будь-якого іншого стану. Тобто для будь-якої літери a, $\delta(q_f, a) = \emptyset$;
- нехай c — число конкатенації регулярного виразу E, s — кількість символів крім дужок — тобто |, *, a і ε. Тоді кількість станів A дорівнює $2s$ — c (лінійне за розміром E);

- кількість переходів, які виходять із будь-якої держави, становить щонайбільше два;
- оскільки NFA з m станів і щонайбільше e переходів з кожного стану може відповідати рядку довжиною n у часі $O(emn)$, NFA Томпсона може здійснювати узгодження шаблонів у лінійному часі, припускаючи алфавіт фіксованого розміру.

Для побудови NDFA необхідно задати граматику мови регулярних виразів та алгоритм, який перетворює вхідну послідовність лексем у обернений польський запис. Обернений польський запис являється ефективним представленням вхідної послідовності, з якою вже зручно оперувати на рівні алгоритму побудови дерева NDFA, а для його побудови використовується метод рекурсивного спуску, який легко реалізовувати та працює з LL(k) граматиками, чим і являється мова регулярних виразів.

В інформатиці рекурсивний аналізатор спуску — це різновид парсера згори вниз, побудований із набору взаємно рекурсивних процедур (або нерекурсивного еквівалента), коли кожна така процедура реалізує один із нетерміналів граматики. Таким чином, структура отриманої програми тісно відображає структуру граматики, яку вона визнає.

Прогностичний аналізатор — це рекурсивний аналізатор спуску, який не потребує зворотного відстеження. Прогнозний синтаксичний аналіз можливий лише для класу граматик $LL(k)$, які є безконтекстними граматиками, для яких існує деяке додатне ціле число k, що дозволяє рекурсивному синтаксичному аналізатору вирішити, яку продукцію використовувати, вивчаючи лише наступні k лексеми k вхід. Отже, граматики $LL(k)$ виключають усі неоднозначні граматики, а також усі граматики, що містять ліву рекурсію. Будь-яка без контексту граматика може бути перетворена в еквівалентну граматику, яка не має лівої рекурсії, але видалення лівої рекурсії не завжди дає лірику $LL(k)$. Прогнозний аналізатор працює в лінійний час.

Рекурсивний спуск із зворотним відстеженням - це техніка, яка визначає, яке виробництво використовувати, намагаючись кожне виробництво по черзі. Рекурсивний спуск із зворотним відстеженням не обмежується граматикою $LL(k)$, але не гарантується, що припиниться, якщо граматика не буде $LL(k)$. Навіть коли вони припиняються, парсерам, які використовують рекурсивний спуск із зворотним відстеженням, може знадобитися експоненційний час.

Хоча прогнозовані парсери широко використовуються і їх часто вибирають, якщо писати аналізатор вручну, програмісти часто вважають за краще використовувати аналізатор на основі таблиці, створений генератором парсера [потрібне цитування], або для мови $LL(k)$, або використовуючи альтернативу аналізатор, такий як LALR або LR. Особливо це стосується, якщо граматика відсутня у формі $LL(k)$, оскільки відбувається перетворення граматики в LL, щоб зробити її придатною для прогнозного розбору. Прогностичні аналізатори також можна автоматично генерувати, використовуючи такі інструменти, як ANTLR.

Прогностичні парсери можуть бути зображені за допомогою діаграм переходу для кожного нетермінального символу, де краї між початковим та кінцевим станами позначені символами (терміналами та нетерміналами) правої частини виробничого правила.

Базовим алгоритмом для мови регулярних виразів являється **Non-deterministic Finite Automaton** (NDFA), або ж недетермінований кінцевий автомат. Недетермінований автомат — це абстрактний автомат, кількість внутрішніх станів якого не нескінченне. У контексті регулярних виразів, NDFA являє собою направлений граф, де вузлами являються стани автомата, а ребра між ними — переходи (букви чи слова). Цей алгоритм досить просто реалізовувати, тому доречно використовувати саме його. Плюсом даного методу являється простота задання шаблону та велика швидкість роботи, однак повноцінна реалізація мови регулярних виразів може стати нетривіальною задачею, і тому, частіше всього, використовуються базові оператори.

Іншим підходом являється пошук по простому шаблону, наприклад, 1 іменник та 1 дієслово. Даний метод за своєю сутністю схожий на попередній метод, однак він менш гнучкий у кількості варіацій можливих концептів, однак його висока швидкодія може бути доречною у випадку обробки великих масивів даних. Також даний метод доречно використовувати для пошуку простих за своєю структурою концептів.

2.4.4 Методи фільтрації цільових сутностей

Фільтрація концептів — важливий етап автоматичного виділення концептів з документа, адже на виході з шукача концептів може бути багато непотрібних варінатів, що буде лише заважати коректній роботі системи, тому постає необхідність у деяких правилах, за якими ключові слова будуть відсікатися.

Частіше всього, відсікають такі ключові слова, кількість яких у документі менше деякого порогового значення (зазвичай 1–5). Також використовують список стоп-слів, при наявності котрих у ключовому слові, воно відсікається від решти. Плюсом даного методу являється простота реалізації, однако разом з простотою необхідно вручну знаходити стоп-слова для формування списку. Критерієм відбору стоп-слів являється їх частина мови (частка, тощо) та ступінь емоцуійної забарвленості. Чим більше забарвлене слово, тим більша вірогідність того, что концепт з даним словом необхідно відфільтрувати.

Фільтрація ключових слів — перший етап виділення важливих концептів з тексту. Фільтрація у найпростішому випадку представляє собою перевірку наявності у ключовому слові деяких стоп-слів. Стоп-слова — це слова, які небажані у ключових словах та які не несуть сенсу для контексту. Цими словами являються сполучники, займенники, тощо. Також у список стоп-слів можуть входити вручну створені списки, специфічні для предметної області та ресурсу документів. Наприклад, для наукових статей такими стоп-словами являються ”Рис”, ”Табл”, ”Тест”, тощо.

Базова фільтрація призначена для грубого відсіювання ключових слів, для більш точного фільтрування необхідно прорангувати кожне ключове слово та відсіяти все, що нижче деякого порогового значення. Існує багато алгоритмів рангування ключових слів, однак найбільш популярним являються TF-DF та BM25.

Алгоритм TF-IDF (term frequency-inverse document frequency) — метод, який рангує важливість ключових слів для загального контексту документа, який враховує частоту ключового слова в документі та частоту ключового слова у всьому корпусі. Існує і інший алгоритм, який статистично працює краще, ніж зазначений вище — BM25, який принципіально не відрізняється від TF-IDF (враховується частота ключового слова у документі та у всьому корпусі).

У пошуку інформації, tf-idf або TFIDF, що скорочується на термін “зворотна частота документа”, є числовою статистикою, яка призначена для відображення того, наскільки важливим є слово для документа в колекції або корпусі. Він часто використовується як коефіцієнт зважування для пошуку інформації, пошуку тексту та моделювання користувача. Значення tf – idf збільшується пропорційно кількості разів, коли слово з’являється в документі, і компенсується кількістю документів у корпусі, що містять слово, що допомагає налаштувати на те, що

деякі слова взагалі з’являються частіше. tf-idf — одна з найпопулярніших схем зважування термінів сьогодні; 83% текстових систем рекомендацій у цифрових бібліотеках використовують tf-idf.

Варіанти схеми зважування tf-idf часто використовуються пошуковими системами як центральний інструмент для оцінки та ранжування релевантності документа з урахуванням запиту користувача. tf-idf може бути успішно використаний для фільтрації слів у різних тематичних полях, включаючи підбиття тексту та класифікацію.

Одну з найпростіших функцій ранжування обчислюють шляхом підсумовування tf-idf для кожного терміна запиту; багато більш складних функцій ранжирування — це варіанти цієї простої моделі.

Припустимо, у нас є набір англійських текстових документів і хочемо класифікувати, який документ найбільше відповідає запиту, “коричнева корова”. Простий спосіб почати - це вилучити документи, які не містять усіх трьох слів “корінь”, “коричневий” і “корова”, але це все ще залишає багато документів. Для подальшого їх розрізнення ми можемо порахувати кількість разів, коли кожен термін виникає в кожному документі; кількість разів, коли термін виникає в документі, називається його частотою. Однак у випадку, коли тривалість документів сильно різниться, часто вносяться корективи (див. Визначення нижче). Перша форма термінового зважування пояснюється Гансом Пітером Луном, який можна узагальнити як: Вага терміна, який зустрічається в документі, просто пропорційний частоті терміна.

Оскільки термін “the” настільки поширений, частота терміна, як правило, неправильно підкреслює документи, які, як правило, використовують слово “the” частіше, не надаючи достатньої ваги більш значущим термінам “коричневий” і “корова”. Термін “the” не є хорошим ключовим словом для розрізнення відповідних та невідповідних документів та термінів, на відміну від менш поширених слів “коричневий” та “корова”. Отже, включений зворотний коефіцієнт частоти документа, який зменшує вагу термінів, які трапляються дуже часто в наборі документів, і збільшує вагу термінів, які трапляються рідко.

У пошуку інформації Okapi BM25 (BM — це абревіатура найкращого узгодження) — це система ранжування, яка використовується пошуковими системами для оцінки відповідності документів певному пошуковому запиту. Він

ґрунтується на ймовірнісних системах пошуку, розроблених у 1970-х та 1980-х роках Стівеном Е. Робертсоном, Карен Спарк Джонсом та іншими.

Назва фактичної функції ранжування - BM25. Більш повна назва Okapi BM25 включає назву першої системи, яка її використовувала, що була системою пошуку інформації Okapi, впровадженою в Лондонському міському університеті у 1980-х та 1990-х роках. BM25 та його новіші варіанти, наприклад BM25F (версія BM25, яка може враховувати структуру документа та прив'язувати текст), являють собою найсучасніші функції пошуку TF-IDF, що використовуються для пошуку документів.

Існують і більш просунуті методи по фільтрації концептів: **TF-IDF** чи **BM25**. Дані методи беруть до уваги частоту потрапляння слова у документ та вірогідність зустріти його у будь-якому документі з корпусу, тобто ранг слова у документі прямо пропорційний його частоті у цьому документі та оберненопропорційний до вірогідності знайти дане слово у деякому документі. **TF-IDF** та **BM25** концептуально не відрізняються, однак **BM25** являється більш якісним алгоритмом, адже він бере до уваги довжину документа, що корегує кінцевий ранг слова.

2.4.5 Ранжування цільових сутностей

Ранжування концептів — один за найважливіших етапів створення алгоритму по автоматичному виділенню концептів, адже саме від нього залежить кінцева якість результатів. Алгоритм ранжування ранжує концепти за деяким критерієм "якості", який обчислюється для кожного концепта та який визначає його важливість для розуміння контексту.

Найпростішим варіантом являється сортування концептів за їх частотою потрапляння у тексті. Даний метод найпростіший в реалізації, однак не враховує морфологічні особливості концепта, що може негативно вплинути на якість. У цього метода існують й інші, більш просунуті варіанти, такі як **TF-IDF** [9], **BM-25** [10] та інші. Вищеназвані методи в оцінці концепта також використовують вірогідність потрапляння концепта у тексті, однако все ще не враховуть морфологічні особливості, а у випадку малої кількості даних можуть видавати неякісні результати.

Іншим підходом являється ранжування концептів за допомогою алгоритму **TextRank** [11] та його похідних. В його основі лежить представлення концептів у вигляді графу, де концепти — вершини, а його ребрами можуть бути різно-

манітні критерії зв’язку між концептами (наявність спільних слів у контекстах, кількість спільних слів у самому контексті, схожість слів за косинусною метрикою відповідних їм Word2Vec векторів, тощо). Далі отриманий граф представляють у вигляді квадратної матриці, яка йде на вхід алгоритму **PageRank** [12], який факторизує матрицю у вектор з рангами концептів. Даний метод має перевагу у вигляді алгоритму, який враховує важливість концептів на основі схожих до нього інших концептів, однак мінусом являється мала стабільність.

2.4.6 Анотування цільових сутностей

Існує декілька підходів щодо анотування цільових сутностей. Перший підхід полягає у максимізації повноти щодо знаходження анотацій [13], тобто знаходить всі речення, в яких потрапляється деяка цільова сутність. Даний підхід є найпростішим із всіх та дає повний обсяг інформації щодо концепту, однак, за умови значного обсягу тексту, може негативно сказитися на якості виділення тез, адже експерту необхідно буде продивитись їх велику кількість.

Другий підхід використовує деякі шаблони, за якими обираються найбільш змістовні речення з концептом. Даний підхід, в залежності від шаблону, здатний якісно виділяти тільки необхідні речення, однак, за умови малого обсягу тексту, даний алгоритм може показати помірні результати та нічного не видати.

Третій підхід заснований на нейронних мережах та здатний автоматично генерувати тезу на основі вхідного тексту та концепту [14]. Даний алгоритм використовує рекурентні нейронні мережі [15] чи нейронні мережі-трансформери [16] для генерації тез. Тренування нейронних мереж здійснюється на великих корпусах тексту, однак також можна використовувати предтреновані варіанти та дотренувати на власних даних, щоб нейронна мережа могла повноцінно працювати з даною предметною областю. Плюсом даного підходу являється можливість створити тезу навіть з малим обсягом даних, однак основним мінусом являється громіздкість підходу, необхідність тренувати нейронну мережу на великих обсягах даних зі спеціальною апаратурою та недетерміновані результати, які проблематично контролювати.

2.5 Висновки по розділу

На сьогодні для текстів дидактичної направленості не існує відкритих програмних комплексів по автоматичному виділенню концептів та тез. Однак для вирішення даної задачі існує велика кількість алгоритмів, композиція з яких здатна розв'язувати поставлену проблему.

Ключовими етапами обробки будь-якого тексту являються токенізація, виділення цільових сутностей, їх фільтрація та ранжування. Етап токенізації, не дивлячись на значне розмаїття методів, являється базовим кроком, і тому він майже не потребує спеціальних модифікацій для роботи з тем чи іншим текстом.

Крок з виділенням цільових сутностей, на відміну від токенізації, має суттєві відмінності у своїх підходах та у множині задач, які цей етап виконує. В основному, задача виділення цільових сутностей розглядається для текстів наукового плану з використанням простих шаблонів, які не охоплюють всі можливі варіанти концептів. У даній роботі розглядаються тексти освітньої направленості, з яких необхідно виділяти цільові сутності за довільним шаблоном будь-якого рівня комплексності, і для вирішення даної задачі підходить мова регулярних виразів. Однак дана мова регулярних виразів, на відміну від класичних реалізацій, повинна працювати на рівні слів та їх атрибутів, і тому постає необхідність у розробці варіанту, який здатен задовольнити поставлені перед ним умови.

Підходи до фільтрації та ранжування цільових сутностей також залежать від предметної області та завдань, які ставляться перед системою. Однак дані методи мають деякі спільні риси, наприклад, врахування частоти слів, їх морфології та позиції у тексті, і саме тому доцільно використати класичні алгоритмічні рішення та запропонувати нові евристичні метрики для ранжування цільових сутностей.

Автоматичне анотування цільових сутностей, у загальному випадку, виражається у двох варіантах роботи з текстом: пошук множини речень, які описують сутність чи абсолютно повністю згенерувати анотацію. Перший підхід обмежений лише кількістю текстових даних на вході, а другий може видавати непередбачувані результати, які можуть не корелювати з цільовою сутністю. Перший підхід доречно використовувати на великих об'ємах даних, адже на виході буде велика кількість кандидатів у анотацію. Тому є необхідність у розробці деякого критерію, за яким будуть фільтруватися отримані кандидати для у анотацію.

3. ФОРМАЛЬНИЙ АПАРАТ ТА МЕТОДИ АВТОМАТИЧНОГО ВИДІЛЕННЯ КОНЦЕПТІВ ТА ТЕЗ

У даній роботі була обрана низка алгоритмів для автоматичного виділення концептів та тез, які крок за кроком оброблюють текст, виділяючи його ключові частини.

Запропонований алгоритм працює з необробленим HTML документом та на виході повертає проранжований список концептів та тез. Діаграма роботи алгоритму зображена на рисунку 3.1.

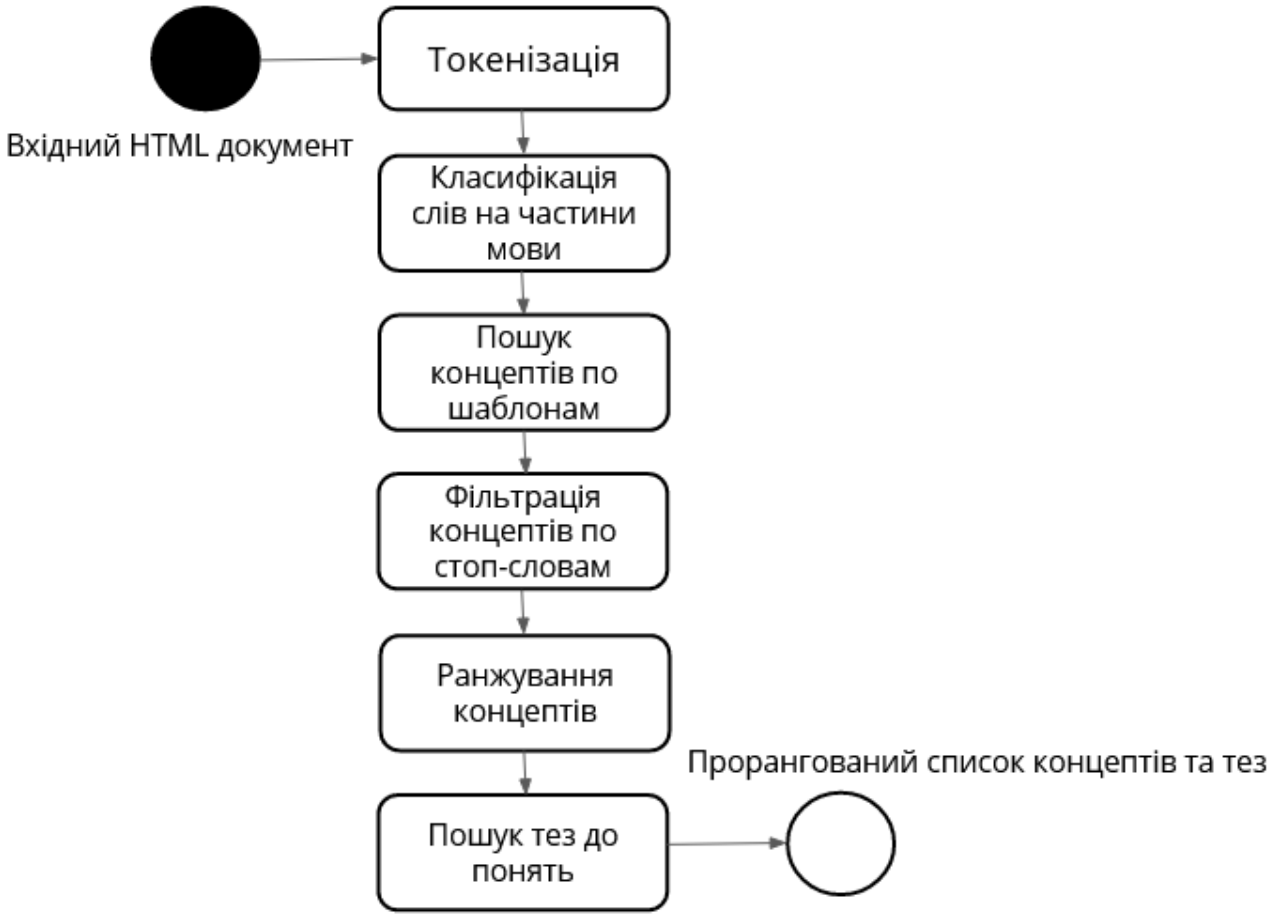

Рисунок 3.1. Діаграма роботи алгоритму автоматичного виділення концептів та тез

Специфіка статей з освітньо-наукового порталу дозволяє активно використовувати структурні одиниці з документа, такі як заголовки, параграфи та анотацію, що дозволяє спростити визначення евристик для ранжування концептів.

3.1 Алгоритм токенізації тексту

Алгоритм токенізації тексту — це початковий та базовий алгоритм, який на вхід приймає корпус тексту, а на виході повертає список токенів.

У даній роботі була обрана реалізація алгоритму токенізації в бібліотеці для обробки та аналізу текстів SpaCy [17]. Запропонований алгоритм базується на гіпотезі, що не існує таких слів, які можуть містити в собі пробіли. Хід роботи алгоритму зображений на рисунку 3.2.

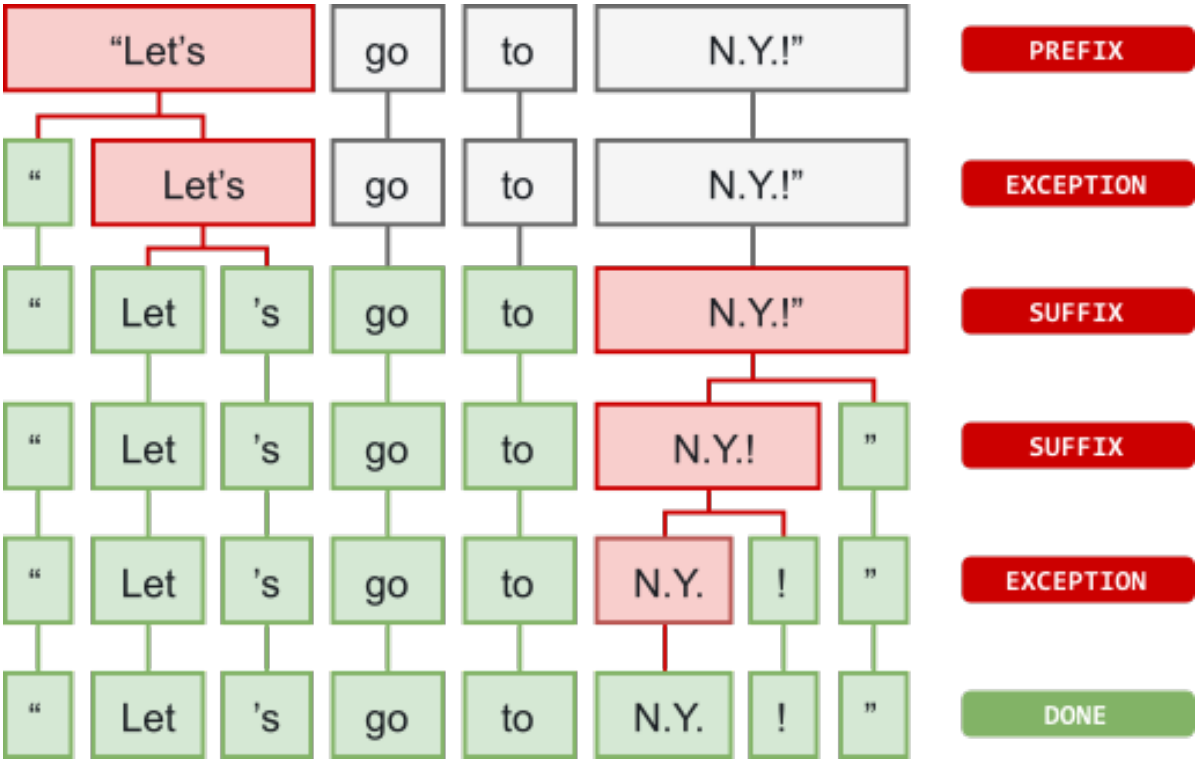

Рисунок 3.2. Діаграма роботи алгоритму токенізації

Даний алгоритм використовує підхід, де для кожного токена, який був виділений за допомогою токенізації по пробілам, проводить більш детальну токенізацію, поступово дроблячи його на більш маленькі токени. Також даний алгоритм замінює всі токени тире на символ дефісу, а всі токени лапок на «”».

Для даного алгоритму були запропоновані наступні модифікації:

- пунктуація всередні слів (дефіси, апострофи) не розбиває слово на токени;
- токени всередині лапок перетворюються в один великий токен.

У даній роботі був використаний саме цей алгоритм токенізації по причині гнучкості його налаштування та високої швидкодії, яка досягається за допомогою мови програмування C.

3.2 Алгоритм класифікації слів на частини мови

Алгоритм класифікації слів на частини мови (Part of Speech Tagging) являється важливим кроком у виділенні концептів, і тому у даній роботі був використаний алгоритм softmax регресії, оскільки цей алгоритм надає велику точність класифікації разом зі значною швидкодією.

Діаграма роботи алгоритму зображена на рисунку 3.3.

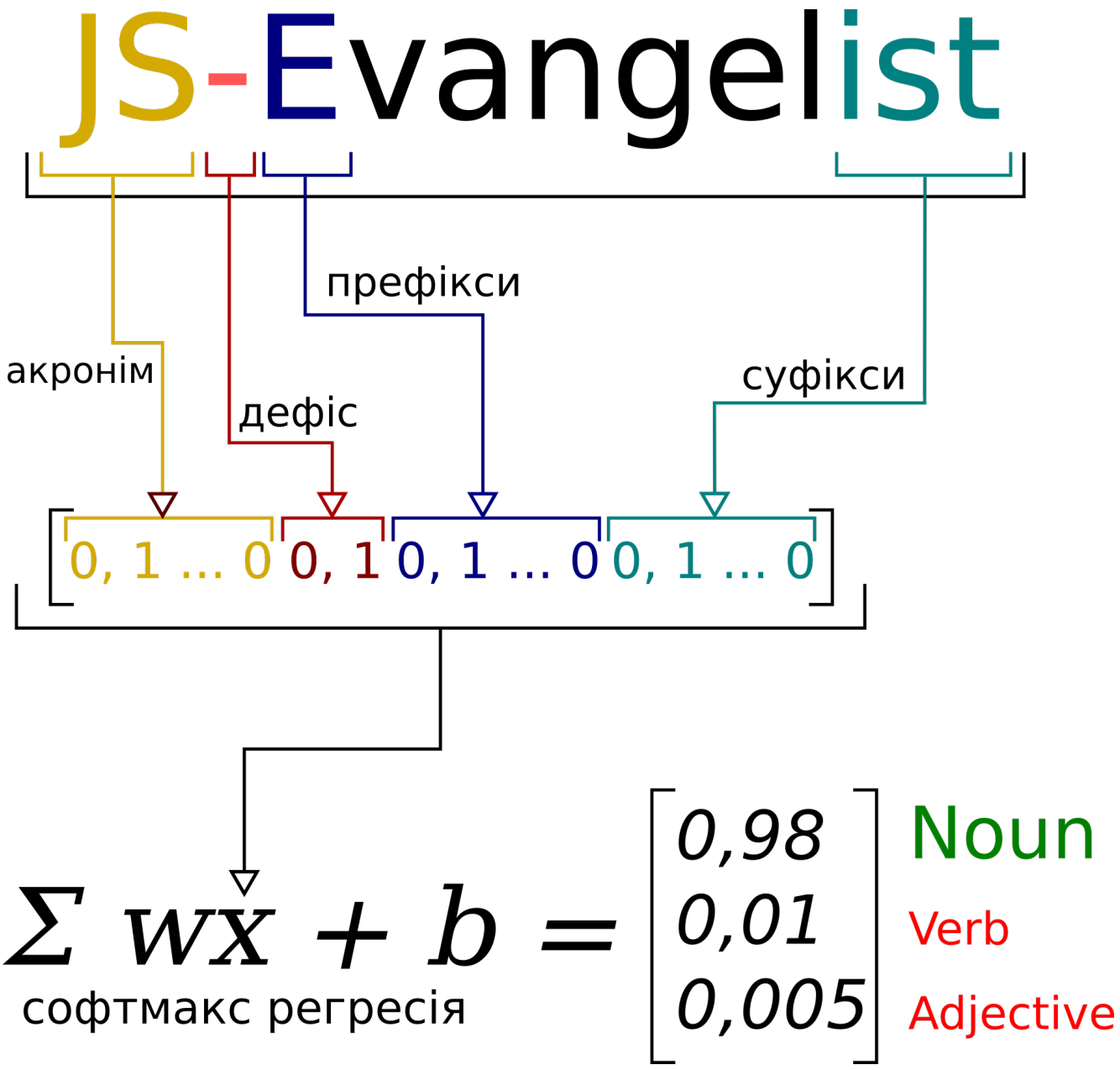

Рисунок 3.3. Діаграма роботи алгоритму класифікації на частини мови

Класифікація на частини мови за допомогою логістичної регресії починається з виведення параметрів, по яким логістична регресія буде класифікувати

слово. Параметрами являється числовий вектор, кожний елемент якого відповідає за деяку ключову особливість слова та його контексту.

Основними параметрами являються:

- “чи є це слово початком речення?” (0 чи 1);
- “чи є це слово кінцем речення?” (0 чи 1);
- довжина слова;
- “чи написане слово з великої літери?” (0 чи 1);
- “чи всі літери у слові прописні?” (0 чи 1);
- “чи має слово дефіс?” (0 чи 1);
- “чи має слово точку?” (0 чи 1);
- “чи має слово справа точку?” (0 чи 1);
- “чи має слово зліва точку?” (0 чи 1);
- аналогічні дії повторюються і для слів справа та зліва (по 2 слова).

Далі отриманий вектор подається на softmax регресію і на виході отримуємо вірогідності належності слова до того чи іншого класу.

Softmax регресія — це узагальнення логістичної регресії для випадку, коли стоїть задача класифікації на декілька класів. У випадку логістичної регресії припускається, що класи мали бінарну структуру: $y(i) \in \{0,1\}$. Softmax регресія дозволяє обробляти $y(i) \in \{1, \ldots, K\}$, де K — кількість класів.

Функція вірогідності визначається наступним чином:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{j_i}}$$

, де z_i — i-й елемент вхідного вектора, який необхідно перетворити у набір вірогідностей. Експонента у формулі необхідна для того, щоб оброблювати негативні значення z_i і для білш простого диференціювання.

Таким чином, у цій роботі для класифікації слів на частини мови був обраний алгоритм softmax регресії, оптимізація вагів була реалізована за допомогою метода Ньютона, а корпус для тренування був скомпонований з ресурсу opencorpora.com. Результуюча модель була протестована на вибірці з 50000 одиницях з тренувальних даних, та точність класифікації показала 97.76%.

3.3 Алгоритм пошуку концептів по заданому регулярному виразу

Для цієї задачі підходить алгоритм під назвою Non-deterministic Finite Automaton (NDFA), або ж недетермінований кінцевий автомат по причині його простоти, ефективності та легкості в реалізації. Недетермінований автомат — це абстрактний автомат, кількість внутрішніх станів якого не нескінченне. NDFA являє собою граф, де кожна вершина — це стан автомату, а переходи між ними у контексті регулярних виразів — символи чи слова, які поступаються після іншого алгоритму парсингу регулярного виразу.

Для побудови NDFA попередньо необхідно перевести регулярний вираз у інверсну польський запис та перетворити його у NDFA за допомогою алгоритму Томпсона. Інверсний польський запис можливо отримати після алгоритму парсингу регулярного виразу, який був реалізований за допомогою рекурсивного спуску.

Алгоритм рекурсивного спуску — це алгоритм для синтаксичного аналізу, який будується за допомгою взаємно рекурсивних процедур (або не рекурсивних еквівалентів), кожна із яких створює одну із одиниць граматики.

Алгоритм Томпсона — це метод для трансформації регулярного виразу в еквівалентний недетермінований скінченний автомат. Даний алгоритм будує NDFA граф з урахуванням операторів регулярних виразів, таких як «+», «?», «*» та дужки.

NDFA зазвичай представляється у вигляді набору з п’яти параметрів: Q, Σ, φ, q_0 та F, де Q — набір станів, Σ — набір символів (алфавіт), φ — функція переходу, q_0 — початковий стан $q \subseteq Q$, F — набір фінальних станів ($F \subseteq Q$). NDFA по своїй суті представляє собою граф, де кожна вершина — це один зі станів, а ребра — переходи між станами, вхідний стан зазвичай зображається на графіках однією вхідною лінією, а кінцевий — подвійним колом.

Нехай нашим недетермінованим кінцевим автоматом є автомат виду:

- $Q = \{a, b, c\}$;
- $\Sigma = \{0, 1\}$;
- $q_0 = \{a\}$;
- $F = \{c\}$.

Тобто у даному автоматі ми маємо три стани $\{a, b, c\}$, алфавіт з двох символів $\{0, 1\}$, з одного початкового та кінцевого стану.

На рисунку 3.4 зображений приклад графу NDFA.

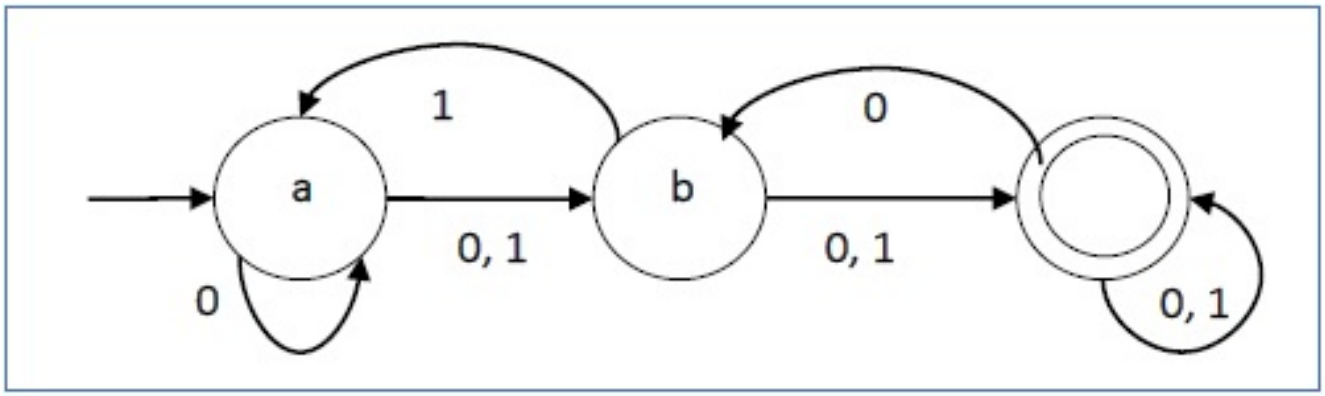

Рисунок 3.4. Приклад NDFA

Даний автомат легко будувати та інтерпретувати та його часто використовують як алгоритм для роботи регулярних виразів у мовах програмування.

Реалізація даного алгоритму для цієї роботи повинна передбачати роботу з частинами мови та зі звичайними регулярними виразами, якими можна задавати вхідні слова. Також цей алгоритм повинен працювати на рівні слів, а не букв, як більшість інших інструментів для регулярних виразів.

Для задання шаблону пошуку ключового слова необхідно створити відповідний діалект для власної реалізації мови регулярних виразів, наприклад, вираз “<pos=ADJ>*<pos=NOUN>+” задає шаблон, при якому знаходяться нуль чи більше прикметників та один чи більше іменників.

В загальному випадку, концепт — це просто послідовність іменників, наприклад “1 чи більше іменників”. Однак цього замало, адже ключові слова також можуть мати деяку якісну характеристику. Тоді шаблон перетворюється у тип “нуль чи більше прикметників та один чи більше іменників”, наприклад, “неймовірна послідовність іменників”. У кінцевому результаті, у цій роботі використовується шаблон “(<pos=ADJ> <pos=NOUN>)* <pos=NOUN>+”.

Таким чином, ключові слова з тексту виділяються за допомогою розробленої мови регулярних виразів, яка базується на алгоритмах NDFA та рекурсивного спуску. Дана мова працює на рівні слів та здатна оперувати з частинами мови слів та зі звичайними регулярними виразами, які задають шаблон слова. Розроблений алгоритм виділяє групи прикметників та іменників, які об’єднані сполучниками, і цей шаблон покриває більшість можливих варіантів ключових слів. Також

цей алгоритм використовується для виділення взаємозв'язків між ключовими словами, однак у даному випадку використовується шаблон для пошуку дієслів та їх форм. Якщо проміжок між ключовими словами повністю співпадає з регулярними виразом для дієслів, то це значить, що система знайшла предикат між двома концептами.

3.4 Алгоритм фільтрації та ранжування концептів

У даній роботі був запропонований метод фільтрації концептів за допомогою стоп-слів та шаблонів на мові регулярних виразів. Фільтрація за стоп-словами необхідний для того, щоб уникати концепти з наступними характеристиками:

- емоційно забарвлені ("жахливий", "чудовий", тощо);
- службові концепти (таблиця, рисунок, чорна лінія на графіку, тощо).

Шаблон для фільтрації концептів враховує контекст концепта, тобто позицію в тексті відностно інших слів. Даний метод фільтрує концепти, які були виділені на попередньому етапі по причині помилки від класифікатора на частини мови. Даний шаблон реалізований на запропонованій вище мові регулярних виразів.

Також запропонований алгоритм фільтрації та ранжування концептів має можливість об'єднувати концепти між собою задля зменшення кількості подібних між собою варіантів у кінцевому результаті. Для цього також використовується шаблон на мові регулярних виразів виду "`<tag=ADJ> <word=w1> <word=w2> ...`", тобто один концепт об'єднується з іншим тоді і лише тоді, коли перший концепт являється підконцептом другого концепта.

Ранжування концептів — важливий етап у виділенні ключових концептів з тексту, адже саме він дозволяє визначатити ступінь важливості концепта для розумення тексту. У метрику для ранжування концептів варто віднести такі параметри як:

- частота концепта у тексті;
- кількість слів у концепті;
- кількість слів з великої літери у концепті;

Вищеназвані характеристики були підібрані евристичними методами, а загальна формула для метрики виглядає наступним чином:

$$score(C) = \frac{boost \times \sqrt{tf} \times max(len(C_t), 1)}{len(C)^0.2}$$

, де — концепт, $boost$ — коефіцієнт, який залежить від позиції концепту в тексті, tf — частота концепту, $len(C_t)$ — кількість слів з великої літери, а $len(C)$ — загальна кількість слів.

Після рангування концептів, необхідно відсікти все, що нижче деякого порогового значення. На виході алгоритму фільтрації система отримує важливі для контексту концепти.

3.5 Висновки по розділу

Задача автоматичного виділення концептів та тез потребує послідовної обробки текстових даних за допомогою множини методів для роботи з текстом. Основними вимогами до алгоритмів являється висока швидкодія та значна якість кінцевого результату для роботи зі статтями з освітньо-наукового порталу. Тому була адаптована та запропонована низка формальних апаратів для вирішення поставленої задачі, яка відповідає заданим вимогам.

Базова обробка тексту починається з етапу токенізації, і для цього був використаний алгоритм, який запропонований у бібліотеці для обробки мов spaCy. Даний алгоритм ітеративно токенізує слова, починаючи з верхнього рівня (пробіли) і закінчуючи розбиттям слів по внутрішній пунктуації слова. Цей підхід зручний тим, що на будь-якому кроці ітерації процес токенізації можна зупинити та отримати токени необхідного рівня розбиття. У даній роботі був запропонований підхід зі збереженням у токенах символів пунктуації (дефіси, апострофи) та з об'єднанням токенів у лапках в один великий токен.

Етап з класифікацією слів на частини мови являється ключовим, адже від якості його роботи повністю залежить фінальний результат. Саме тому був обраний метод, який поєднує в собі високу точність класифікації та значну швидкодію, а саме класифікація на основі морфологічних ознак слова та його контексту. Власне класифікація здійснюється алгоритмом softmax регресії, який був

попередньо натренований на відкритих даних з вручну проставленими частинами мови.

Проставлені частини мови дають можливість виділяти концепти по деякому шаблону, який не обмежений одними лише словами. У даній роботі була запропонована мова регулярних виразів, яка працює на рівні токенів та їх атрибутів. До атрибутів належать частина мови, стем та лемма слова. У запропонованому методі був використаний алгоритм рекурсивного спуску для приведення шаблону до інверсного польського запису з урахуванням базового синтаксису мови регулярних виразів, а для пошуку шаблонів використовується недетермінований скінченний автомат, який будується за допомогою алгоритму Томпсона.

Процес фільтрації та ранжування виділених концептів необхідний для відбору тих одиниць, які найбільш повно покривають тематику текстового документа. У випадку з фільтрацією був обраний метод на основі стоп-слів та шаблонів мовою регулярних виразів, які враховують контекст концепту. Для ранжування концептів була запропонована метрика, яка враховує частоту концепту у документі, кількість слів та морфологічні особливості слів.

Останнім етапом являється пошук тез, який здійснюється також за допомогою запропонованої мови регулярних виразів. На даному кроці для обраного концепту з усіх речень у тексті відфільтровуються ті, які задовільняють заданому шаблону, в які входить обраний концепт.

4. РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ АВТОМАТИЧНОГО ВИДІЛЕННЯ КОНЦЕПТІВ ТА ТЕЗ

4.1 Середовища розробки

Основним засобом створення програмного продукту були інструменти PyCharm 2019.3.4. Для створення веб-додатку використовувався редактор Sublime Text.

4.1.1 Середовище розробки PyCharm

Середовище розробки PyCharm — це середовище розробки для інтерпретованої мови програмування Python. Це середовище забезпечує інструменти для аналізу коду, графічний відладчик pdb, інструменти для тестування та підтримує розробку веб інструментів на Flask. Інструмент PyCharm розроблений компанією JetBrains на основі редактора IntelliJ IDEA. PyCharm може працювати під керуванням операційних систем Windows, Mac OS X і Linux та має наступні функції:

- аналіз кодової бази, підсвічування коду та синтаксичних помилок;
- додаткова навігація між проектами та їх кодом: відображення структури файлів у проеках, швидка навігація між файлами, класами та методами;
- рефакторинг коду та перейменування змінних;
- інструменти для веб-розробки, що використовують у Django;
- вбудований відладчик pdb для Python;
- вбудовані інструменти для юніт-тестування;
- розроблено за допомогою Google App Engine;
- повна підтримка систем керування версіями проекту: інтерфейс для Mercurial, Git, Subversion, Perforce та CVS з підтримкою змін і злиття.

По причині багатого функціоналу та можливостей, саме це середовище розробки буле обране для розробки цього проекту.

4.1.2 Текстовий редактор Sublime Text 3

Sublime Text — це текстовий редактор із інтерфейсом для прикладного програмування на мові Python та JavaScript. Він здатен підтримувати багато мов програмування та мов розмітки, користувачі здатні додавати функції за

допомогою власноруч написаних плагінів, які зазвичай створюються спільнотою та підтримуються під ліцензією на безкоштовне програмне забезпечення.

Далі наведено список основних особливостей Sublime Text:

- швидка та зручна навігація по файлам та директоріям;
- значна кількість гарячих клавіш;
- підтримка великої кількості плагінів на Python;
- редактор повністю доступний для всіх основних операційних систем.

Даний редактор використовується як основний засіб розробки ти дизайну веб сервісів та для роботи з проектами з великою кількістю файлів.

4.2 Мова програмування та використані бібліотеки

Мовою програмування у даній роботі стала мова Python з набором бібліотек для лінійної алгебри, обробки мов та для створення веб-додатків з можливістю REST інтеграції.

4.2.1 Python

Python — це високорівнева мова програмування загального призначення, яка орієнтована на покращення продуктивності розробника і читання коду. Синтаксис ядра Python досить мінімалістичний. Однак у той же час стандартна бібліотека мови програмування включає великий обсяг корисних функцій.

Python підтримує декілька парадигм програмування, в тому числі структурний, об’єктно-орієнтований, функціональний, імперативний і аспектно-орієнтований. Основні архітектурні риси — це динамічна типізація, автоматичне керування пам’яттю (Garbage collector), інтроспекція, а також механізм обробки виключень, якісна підтримка багатопоточних обчислень й зручні високорівневі структури даних для будь-яких цілей.

4.2.2 Бібліотека spaCy для роботи з англійською мовою

Бібліотека spaCy — це бібліотека програмного забезпечення з відкритим кодом для розширеної обробки мов, написана на мовах програмування Python та Cython. Бібліотека видається за ліцензією MIT і в даний час пропонує статистичні моделі та нейронноі мережі для англійської, німецької, іспанської, португальської,

французької, італійської, голландської та багатомовної NER, а також токенізація для різних інших мов.

На відміну від NLTK, який широко використовується для викладання та досліджень, spaCy зосереджується на наданні програмного забезпечення для цілей виробництва. Станом на версію 1.0, spaCy також підтримує deep learning, які дозволяють підключати статистичні моделі, що навчаються популярними бібліотеками машинного навчання, такими як TensorFlow, Keras, Scikit-learn або PyTorch. Бібліотека машинного навчання для spaCy, Thinc, також доступна як окрема бібліотека з відкритим кодом Python. У ньому представлені згорткові моделі нейронної мережі для тегування на частини мови, розбору залежностей та розпізнавання іменованих сутностей, а також удосконаленно API навколо моделей навчання та оновлення та побудови індивідуальних конвеєрів обробки.

Бібліотека spaCy являється стандартом для автоматичного аналізу та обробки текстів для всіх мов європейської групи (англійська, німецька, російська, тощо), тому у даній роботі для роботи з текстом була обрана саме ця бібліотека.

4.2.3 Мікрофреймворк Flask Для розробки веб-додатків

Мікрофреймворк Flask — це фреймворк, який написаний на Python. Він класифікується як мікрофреймворк, оскільки не потребує конкретних інструментів чи бібліотек. У нього немає шару абстрагування бази даних, перевірки форм або будь-яких інших компонентів, де раніше існуючі сторонні бібліотеки забезпечують загальні функції. Однак Flask підтримує розширення, які можуть додавати функції програми так, ніби вони були реалізовані в самій Flask. Розширення існують для ORM, перевірки форм, обробки завантажень, різних технологій відкритої аутентифікації та декількох загальних інструментів, пов'язаних із рамками. Розширення оновлюються набагато частіше, ніж основна програма Flask.

До веб-сервісів, що використовують рамку Flask, належать Pinterest, LinkedIn, та веб-сторінка спільноти для самої Flask.

Flask був створений Арміном Ронахером з Pocoo, міжнародною групою ентузіастів Python, утвореною в 2004 році. За словами Ронахера, ідея спочатку була жартівливим анекдотом, який був досить популярним, щоб зробити серйозну заяву.

Незважаючи на відсутність великого випуску, Flask став популярним серед любителів Python. Станом на середину 2016 року це була найпопулярніша

бібліотека для веб-розробок Python на GitHub та була визнана найпопулярнішим веб-фреймворком в опитуванні розробників Python 2018.

Мікрофреймворк Flask легким та простим інструментов для створення як веб-додатків, так і REST API додатків завдяки своїй багатій екосистемі зі сторонніх бібліотек, саме тому у даній роботі був використаний саме цей фреймворк.

4.3 Система контейнеризації

Інструмент для контейнеризації Docker — це інструмент для керування окремими контейнерами у операційній системі Linux, Windows, MacOS. Docker створює набір інструментів LXC для API-інтерфейсів більш високого рівня, що дозволяє керувати контейнерами на рівні окремих процесів. Також механізм контейнеризації Docker дозволяє, не беручи до уваги вміст контейнера, запускати різноманітні процеси у повному режимі ізоляції, а потім передавати чи клонувати контейнерні системи, створені для цих процесів, на інші сервіси, беручи на себе всі роботи з створення та підтримки контейнерів.

Основні можливості Докер:

- ізоляція всіх ресурсів: споживання системних ресурсів, таких як споживання пам’яті та завантаження центрального процесору, може бути обмежено індивідуально для кожного контейнера за допомогою груп;
- можливість створювати контейнери, що містять складні програмні стеки, шляхом об’єднання існуючих контейнерів, що містять компоненти формуваного стека. Зв’язування здійснюється не через об’єднання вмісту, а через взаємозв’язок між контейнерами (створюється мережевий тунель).
- використання легких контейнерів (які будуються на основі контейнеризації) для ізоляції процесів від інших процесів та основної системи;
- ізоляція на рівні FS системи: кожний процес виконується в повністю окремому root середовищі;
- можливість розміщення в ізольованому файловому просторі неоднорідного source коду, який включаэ в себе різноманітні комбінації файлів, які можуть запускатися, бібліотек, конфігураційних файлів, скриптів, файлів у системах баз даних;

- підтримка роботи на будь-якому сервері повністю основі архітектури x86_64 з системою Linux, починаючи від ноутбуків до серверів та віртуальних машин. Можливість працювати на сучасних та старих ядрах Linux і в стандартних середовищах всіх основних дистрибутивів Linux;

Таким чином, для запуску сервісу для виділення концептів та тез був обраний інструмент Docker для зручного розгортання сервісу у будь-якому середовищі.

4.4 Опис архітектури та реалізації системи для автоматичного виділення концептів та тез

Система для автоматичного виділення концептів і тез — це програмний комплекс, який складається с багатьох модулів, які залежать один від одного та утворюють складу структуру, яка на виході видає готовий список концуептів.

4.4.1 DFD діаграма системи

Діаграма потоку даних (DFD) — модель проектування, графічне представлення ”потоків” даних в інформаційній системі. Діаграма потоків даних також може використовуватись для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму потоків даних рівня контексту, завдяки чому буде показано взаємодію системи із зовнішніми модулями. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати розлого розроблювану систему. Діаграми потоків даних містять чотири типи графічних елементів: процеси, сховища даних, зовнішні ресурси та потоки даних міє елементами діаграми.

На даній (рисунок 4.1) діаграмі система працює з базою даних навчального порталу, на якій знаходится таблиця зі статтями, по яким алгоритм й буде знаходити концепти та тези. Отримані концепти та тези далі рекомендуються екперту х онтологіх, він їх редагує, фільтрує та додає у базу даних разом з іншими концептами, які він власноруч створив у процесі вивчення статті. Збережені концептів у подальшій роботі системи створюбть граф концептів та може бути корисним для візуалізації предметної області.

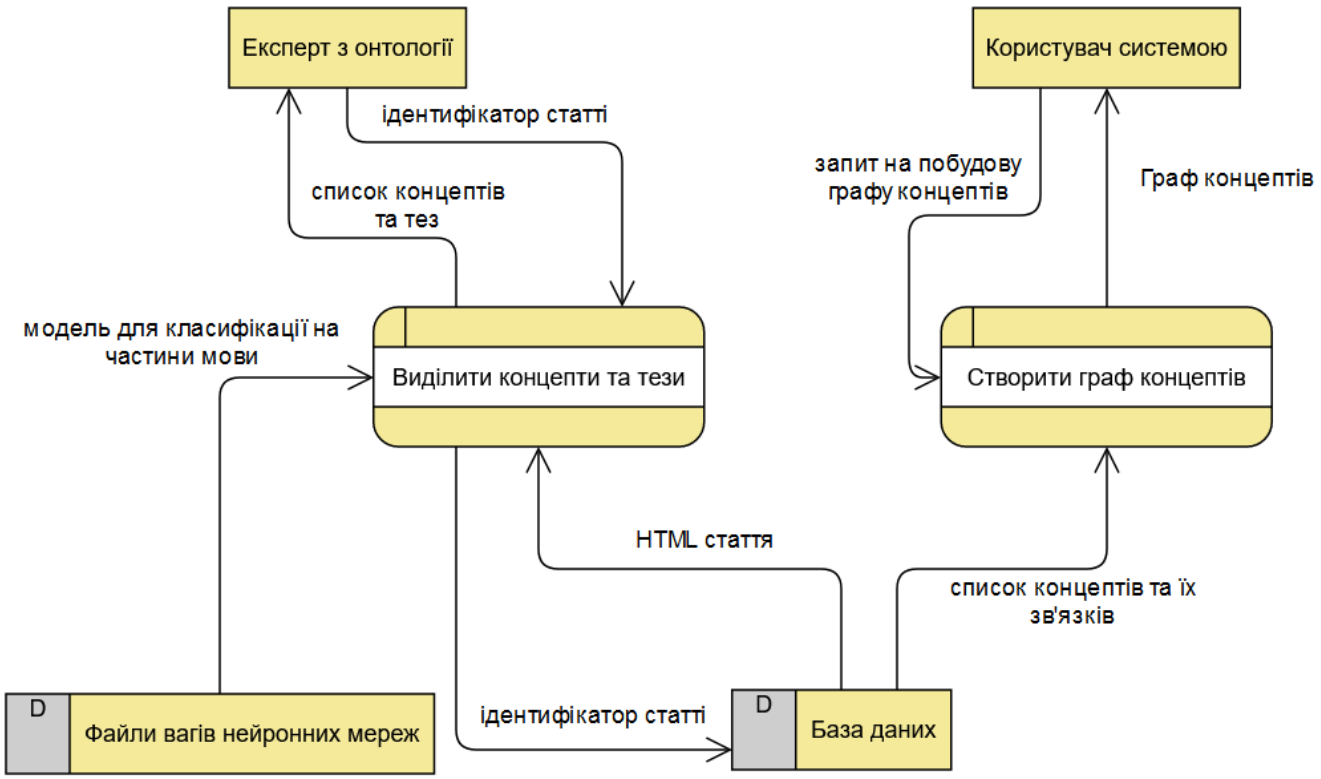

Рисунок 4.1. DFD діаграма системи

У даній роботі DFD діаграма відображає процес видылення концептів зі стстей, тобто користувач надає системі посилання на запис у таблиці зі статтями. Далі система оброблює цей запис, проводить через всі етапи алгоритму та видає на виході готовий список концептів.

4.4.2 Діаграма прецедентів

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання та всі можливості актора користування системою.

Суть даної (рисунок 4.2) діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Експерт з онтології здатен додати власноруч концепт з його тезами у базу даних, а також здатен зробити запит до системи для пошуку нових концептів.

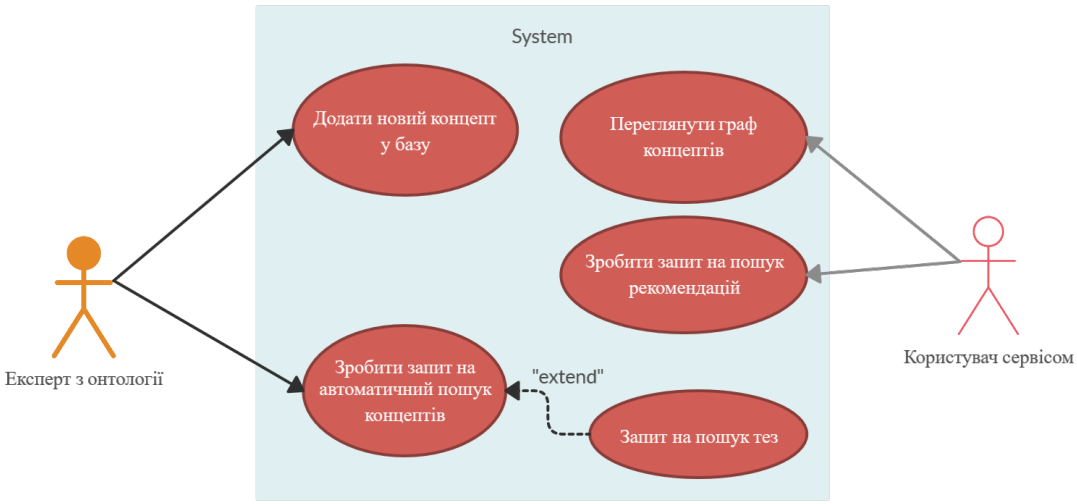

Рисунок 4.2. Use case діаграма системи

Згідно з цією діаграмою, користувач системи (експерт з онтології) має можливість додавати власноруч створені концепти чи робити запит до системи за автоматично виділеними концептам, які також можуть бути доповнені тезами.

4.4.3 REST API інтерфейс системи

Інтерфейс REST API системи представляє собою веб-сервіс, який запускається на виділеному сервері чи на персональному комп'ютері, який підключається до бази даних навчального порталу. Сервіс працює під операційною системою Linux на німецькому хостинг провайдері Hetzner, а власне REST API система працюють у системі контейнеризації Docker, що дозволяє повністю ізолювати робочий простір програми від робочої системи, на якій вона запускається.

REST API додаткок був створений за допомогою мікрофреймворку для створення веб-сервісів Flask з бібліотекою Flask-Restful, яка довзоляє задавати точки доступу до API у ООП вигляді. Також був використаний інструмент для розпаралелення обробки запитів gunicorn, яка створює декілька процесів з Python сервером. У якості фронтенду для доступу до сервісту був обраний веб-сервер Nginx, який дозволяє за допомогою конфігураційних файлів задавати логіку доступу до різних сервісів чи до статичних файлів.

REST API інтерфейс системи складається з одієї точки доступу (endpoint), `/article/<article_id>/concepts`, який може приймати наступні параметри:

- `lemmatize` $\{0, 1\}$ — параметр, який відповідає за лематизацію концепта;
- `topn` $[1, \infty)$ — параметр, який відповідає за кількість концептів на виході;
- `theses` $\{0, 1\}$ — параметр, який відповідає за знаходження тез до концептів;

- `min_tf` $[1, \infty)$ — параметр, який відповідає за частоту концепта в тексті, нижче якої всі концепти відсікаються.

Приклад запиту до REST API сервісу зображений на рисунку 4.3.

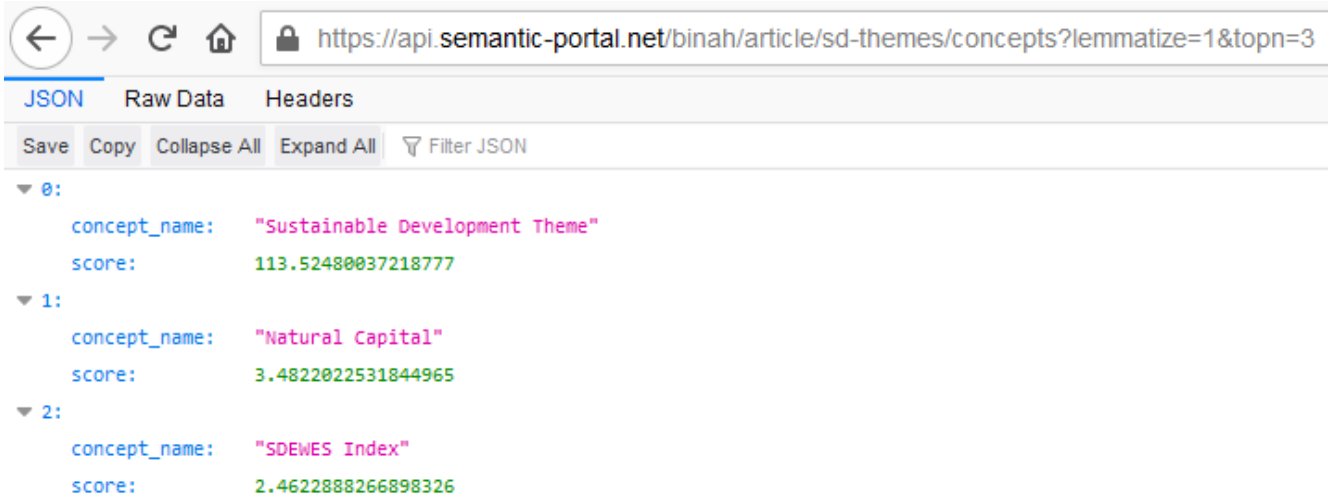

Рисунок 4.3. Запит до сервісу з параметрами `min_tf` та `lemmatize`

4.4.4 Веб-інтерфейс користувача системою

Веб-інтерфейс користувача системою представляє собою сторінку, де він здатен створити запит на виділення концептів та тез з бази даних навчального порталу, задати необхідні параметри та продивитися отримані результати роботи алгоритму.

Веб-інтерфейс для роботи з сервісом був створений за допомогою React та фреймворку для дизайну веб-сторінок Twitter Bootstrap, який дозволяє створювати адаптивні веб-сторінки для всіх форматів екранів. Також був використана бібліотека React-RESTful для комунікації по REST API протоколу з сервісом.

Веб-інтерфейс системи складається з чотирьох складових:

а) Поле для задання статті.
б) Поле для параметрів (`topn`, `lemmatize`).
в) Флаг для підключення ресурсів Вікіпедії.
г) Список отриманих концептів

Сторінка з результатом роботи алгоритму представлена на рисунку 4.4.

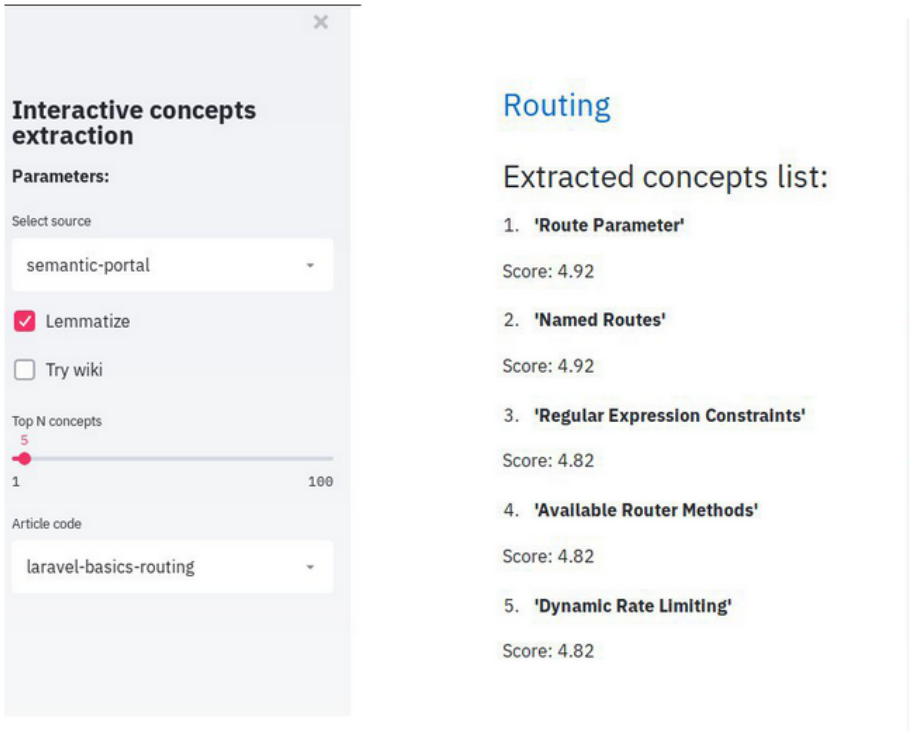

Рисунок 4.4. Веб-інтерфейс з результатом роботи алгоритму для онтологічно-орієнтованого порталу

Також розроблений веб-інтерфейс здатен оброблювати посилання на веб-сторінки з текстовими даними. Сервіс виділяє текст та проводить аналогічні операції, як і у випадку з базою даних. Сторінка з результатом обробки посилання на веб-сторінку представлена на рисунку 4.5.

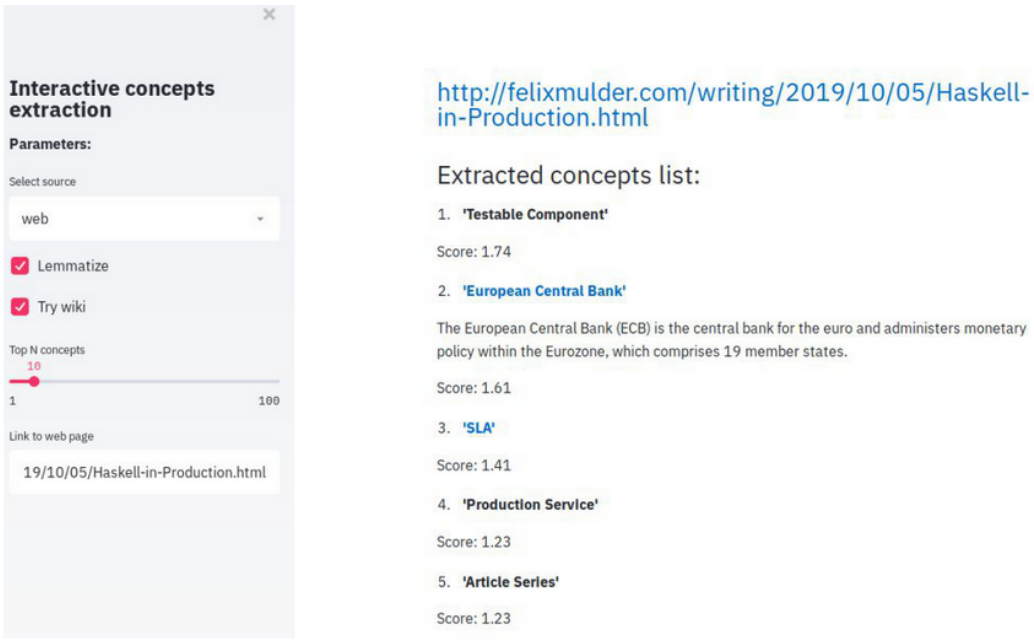

Рисунок 4.5. Веб-інтерфейс з результатом роботи алгоритму для посилання на веб-сторінку

4.5 Висновки по розділу

У даній роботі був розроблений програмний комплекс для обробки текстів зі статей, які надходять або з бази даних освітньо-наукового порталу, або с посилання на веб-сторінку, з якої в автоматичному порядку виділяється текст з корисною інформацією для подальшої обробки. Дана робота була реалізована на мові програмування Python з набором бібліотек для обробки англійської мови та для створення веб-застосунків для реалізації REST API інтерфейсу та інтерфейсу користувача.

Комунікація з сервісом виконується через два канали: REST API чи інтерфейс користувача. Перший канал можна використовувати для інтеграції сервісу з іншими веб-застосунками (онтологічно-орієнтований портал), а другий для тестування роботи системи та для аналізу отриманих результатів.

5. РОЗРОБКА СТАРТАПУ

5.1 Опис ідеї проекту

Постала потреба створити автоматичну систему виділення концептів та тез, на меті якої є допомого експертам з онтології у її побудові. На таблиці 5.1 зображений опис проекту та застосування його можливостей.

Таблиця 5.1. Опис ідеї проекту та його застосування

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Автоматичне виділення концептів та тез без участі експертів	Виділення концептів та тез	Експерт може продавлятись рекомендаціх концептів
	Оптимізація наповнення онтології	Клієнт скоріше отримує нову інформацію
	Пошук рекомендацій	Користувачі здатні отримувати більше статей

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від чинних аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

- визначення переліку техніко-економічних властивостей та характеристик ідеї;
- визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;
- проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають:

а) гірші значення (W, слабкі).
б) аналогічні (N, нейтральні).
в) кращі значення (S, сильні).

Дані характеристики зображені на таблиці 5.2.

Таблиця 5.2. Характеристики проекту

№	Техніко-економічні характе-ристики ідеї	Мій проект	Конкурент 1	Конкурент 2	W	N	S
1	Економічні	Вартість об-слуговування 1000$	Вартість об-слуговування 1200$	Вартість об-слуговування 3000$	3	2	1
2	Надійності	Висока	Низька	Середня			1
3	Технологічні	Середня	Висока	Низька		1	
4	Естетичні	Висока	Середня	Висока			1

5.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару). Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 5.3):

- за якою технологією буде виготовлено товар згідно ідеї проекту?
- чи існують такі технології, чи їх потрібно розробити/доробити?
- чи доступні такі технології авторам проекту?

Таблиця 5.3. Технологічний аудит

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Автоматичне виділення концептів	Використання ПТМ	наявна	доступна
2	Можливість використання	Мова програмування Python	наявна	доступна
3	Створення рекомендаційної системи	Мова програмування Python	наявна	доступна
4	Розгортка серверу	Hetzner	наявна	доступна
5	CI/CD	CircleCI	наявна	доступна

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Проведемо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку. Отримані результати зображені на таблиці 5.4.

Таблиця 5.4. Аналіз попиту

№	Показник стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	5
2	Загальний обсяг продаж, грн/ум.од	100 000$
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	40%

Визначимо потенційні групи клієнтів, їх характеристики, та сформуємо орієнтовний перелік вимог до товару для кожної групи на таблиці 5.5.

Таблиця 5.5. Аналіз аудиторії

№	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба в атоматизованому виділенні концептів	Студенти, навчальні заклади, ІТ-компанії	Ціна для студентів та ІТ-компаній буде різною	Рекомендації щодо відбору концептів

Проводемо аналіз ринкового середовища: складемо таблиці факторів, що сприяють ринковому впровадженню проекту на Табл 5.6, та факторів, що йому перешкоджають на таблиці 5.7.

Таблиця 5.6. Аналіз факторів сприяння

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Зробити виділення концептів автоматичним	Дозволить скоротити час на підготовку навчальних матеріалів	Необхідно виділити більше ресурсів для розробки
2	Модифікації алгоритму на інші предметні області	Дозволить збільшити кількість документів	Вихід на більш крупний ринок та можливо розгортання окремого стартапу
3	Якість	Дозволить залучити деяку кількість аудиторії конкурентів	Дозволить перетворити конкуренцію на монополістичну
4	Сертифікації системи	Зробити систему стандартом перевірки знань в деякиї країнах	Необхідність отримання патенту

Таблиця 5.7. Аналіз факторів сприяння

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Час	Витратити багато часу на побудову алгоритму автоматичного виділення понять, тез та їх класифікацію	Необхідно провести гарний аналіз існуючиї рішень
2	Технологія	Довгій вибір технології автоматичного класифікування понять та тез	Залучення талановитих працівників
3	Якість	Отримання не дуже якіснх тестів	Зосередження на удосконаленні алгоритмів
4	Ринок	Розробки більш універсальних шаблонів для тестування конкурентами	Необхідно запатентувати дану технологію

Проводемо аналіз пропозиції: визначимо загальні риси конкуренції на ринку на таблиці 5.8.

Таблиця 5.8. Аналіз пропозації

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції	Монополістична конкуренція	Монополія дозволить трохи завищити ціну
Рівень конкурентної боротьби	Національна	Необхідно прикласти належні зусилля для охоплення всього національного ринку
Галузева ознака	Внутрішньо-галузева	Необхідно зосередити зусилля на пошуку конкурентних переваг, які дозволять компанії займати стійкі конкурентні позиції на даному ринку
Конкуренція за видами товарів	Товарно-видова	Конкуренція між іншими системами тестування
Характер конкурентних переваг	Не цінова	Головною конкурентною перевагою є якість та швидкість системи
Інтенсивність	Марочна	Диференціація глазурованих сирків за мотивом задоволення

Далі необхідно проаналізувати конкуренцію в галузі за моделлю М. Портера на таблиці 5.9.

Таблиця 5.9. Таблиця Портера

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Клієнти	Товари-замінники
	iSpring, Indigo	Let's test, Testograf	Клієнтам необхідна система автоматичного контролю знань, при цьому основна увага приділяється якості тестів	Товарів замінників багато, але прямих не існує
Висновки:	Прямих конкурентів мало, конкуренція не дуже інтенсивна	Є усі можливості входу на ринок, строк виходу невеликий	Клієнти диктують умови по якості завдань	Обмеження для роботи на ринку незначні

На основі аналізу конкуренції, проведеного (таблиця 5.9), а також із урахуванням характеристик ідеї проекту (таблиця 5.1), вимог споживачів до товару (таблиця 5.4) та факторів маркетингового середовища визначимо та обґрунтуємо перелік факторів конкурентоспроможності на таблиці 5.10.

Таблиця 5.10. Фактори конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Якість концептів	Система створює якісні концепти порівнянно з іншими рішеннями
2	Рекомендаційна система	Рекомендації щодо додавання тих чи інших концептів
3	Легка інтреграція	Можливість підключення системи як частину до іншої
4	Невеликі часові витрати для автоматичного виділення концептів	Концепти виділяються автоматично на відміну від інших рішень

Підіб’ємо підсумки ринкових можливостей за допомогою SWOT аналізу на рисунку 5.1.

Рисунок 5.1. SWOT діаграма

5.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів зображений на таблиці 5.11.

Таблиця 5.11. Опис цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Студенти	Висока	Низький	Низька	Висока
2	Навчальні заклади	Середня	Середній	Низька	Середня
3	ІТ-компанії	Середня	Високий	Середня	Низька

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект. Дана стратегія зображена на таблиці 5.12.

Таблиця 5.12. Стратегія позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Висока якість концептів	Покращувати мовленневі моделі	Простота інтерфейсу	Сервіс, який здатний надати якісну допомогу експерту за онтології.
2	Висока якість тез	Покращувати мовленневі моделі	Повністю вичерані тези	Сервіс, який здатний надати якісну допомогу експерту за онтології у процесі знаходження тез.

5.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 5.13 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.13. Стратегія позиціонування

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Сервіс для автоматичного пошуку конептів та тез до них	Можливість для автоматичного виділення концептів та тез	Можливість інтеграції з іншими сервісами, висока якість моделей

5.6 Висновки по розділу

У даному розділі був розглянутий процес розробки стартап проекту для сервісу по автоматичному виділенню концептів та тез до них для рекомендацій для експерта с онтології. Були розглянуті ризики проекту, його плюси та мінуси, конкурентоспроможність та аналіз цільової аудиторії, яка буде споживати продукт.

ВИСНОВКИ

У даній роботі були досліджені методи та формальні апарати для автоматичного виділення концептів та тез для задачі автоматичної обробки текстів. Оскільки основна маса алгоритмів існує лише для текстів наукових направленостей, то у даній роботі були адаптовані чи запропоновані нові алгоритми спеціально для текстів освітньо-наукової направленості. У число адаптованих алгоритмів входять токенізація тексту, класифікація слів на частини мови та фільтрація отриманих концептів. До запропонованих алгоритмів входять мова регулярних виразів, яка працює на рівні слів та їх атрибутів, метрика для ранжування концептів за їх значущістю для розуміння тематики тексту, а також метод для знаходження тез до концептів. Представлені методики роботи з текстовими даними на виході повертають готовий список знайдених концептів з тезами.

Для реалізації системи для автоматичного виділення концептів та тез був створений веб-застосунок мовою програмування Python з набором бібліотек для роботи з текстом, багатовимірними матрицями, та для створення веб-сервісів. Розроблена система здатна обробляти записи з бази даних онтологічно-орієнтованого порталу та з посилання на веб-сторінку. Сервіс був реалізований на REST API протоколі, тому будь-який інший сервіс здатний інтегруватися з даним веб-додатком та використовувати його можливості. Також був розроблений веб-інтерфейс користувача, який дозволяє переглядати та аналізувати список концептів та тез з бази даний онтологічно-орієнтованого порталу.

СПИСОК ЛІТЕРАТУРИ

[1] Tytenko S. Побудова дидактичної онтології на основі аналізу елементів понятійно-тезисної моделі // Наукові вісті “НТУУ КПІ”. — 2010. — no. 1. — P. 69.

[2] Tytenko S. Програмне забезпечення онтологічно-орієнтованої системи керування інформаційно-навчальним web-контентом // Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 0. — 2011. — P. 78–107.

[3] Tytenko S., Gagarin A. Семантична модель знань для цілей організації контролю знань у навчальній системі // Сборник трудов международной конференции «Интеллектуальный анализ информации-2006».–Київ: Просвіта. — 2006. — P. 298–307.

[4] Lopez P., Romary L. Humb: Automatic key term extraction from scientific articles in grobid // Proceedings of the 5th international workshop on semantic evaluation / Association for Computational Linguistics. — 2010. — P. 248–251.

[5] Martini A., Cardillo A., Rios P. D. L. Entropic selection of concepts unveils hidden topics in documents corpora // arXiv preprint arXiv:1705.06510. — 2017.

[6] Constantin A. Automatic structure and keyphrase analysis of scientific publications : Ph. D. thesis / Alexandru Constantin ; The University of Manchester (United Kingdom). — 2014.

[7] McCandless M., Hatcher E., Gospodnetic O. Lucene in action: covers Apache Lucene 3.0. — Manning Publications Co., 2010.

[8] Kupiec J. Robust part-of-speech tagging using a hidden markov model // Computer Speech & Language. — 1992. — Vol. 6, no. 3. — P. 225–242.

[9] Yun-tao Z., Ling G., Yong-cheng W. An improved tf-idf approach for text classification // Journal of Zhejiang University-Science A. — 2005. — Vol. 6, no. 1. — P. 49–55.

[10] Robertson S., Zaragoza H., Taylor M. Simple bm25 extension to multiple weighted fields // Proceedings of the thirteenth ACM international conference on Information and knowledge management / ACM. — 2004. — P. 42–49.

[11] Mihalcea R., Tarau P. Textrank: Bringing order into text // Proceedings of the 2004 conference on empirical methods in natural language processing. — 2004. — P. 404–411.

[12] Haveliwala T. H. Topic-sensitive pagerank // Proceedings of the 11th international conference on World Wide Web / ACM. — 2002. — P. 517–526.

[13] Gormley C., Tong Z. Elasticsearch: the definitive guide: a distributed real-time search and analytics engine. — " O'Reilly Media, Inc.", 2015.

[14] Rush A. M., Chopra S., Weston J. A neural attention model for abstractive sentence summarization // arXiv preprint arXiv:1509.00685. — 2015.

[15] Schuster M., Paliwal K. K. Bidirectional recurrent neural networks // IEEE Transactions on Signal Processing. — 1997. — Vol. 45, no. 11. — P. 2673–2681.

[16] Attention is all you need / Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit // Advances in neural information processing systems. — 2017. — P. 5998–6008.

[17] Al Omran F. N. A., Treude C. Choosing an nlp library for analyzing software documentation: a systematic literature review and a series of experiments // Proceedings of the 14th International Conference on Mining Software Repositories / IEEE Press. — 2017. — P. 187–197.

[18] Distributed representations of words and phrases and their compositionality / Tomas Mikolov, Ilya Sutskever, Kai Chen et al. // Advances in neural information processing systems. — 2013. — P. 3111–3119.

[19] Pennington J., Socher R., Manning C. D. Glove: Global vectors for word representation // Empirical Methods in Natural Language Processing (EMNLP). — 2014. — P. 1532–1543. — Access mode: http://www.aclweb.org/anthology/D14-1162.

[20] Castelli V., Thomasian A., Li C.-S. Csvd: Clustering and singular value decomposition // IEEE Transactions on knowledge and data engineering. — 2003. — Vol. 15, no. 3. — P. 671–685.

[21] Brants T. Tnt: a statistical part-of-speech tagger // Proceedings of the sixth conference on Applied natural language processing / Association for Computational Linguistics. — 2000. — P. 224–231.

[22] Thompson K. Programming techniques: Regular expression search algorithm // Communications of the ACM. — 1968. — Vol. 11, no. 6. — P. 419–422.

[23] Runeson P., Alexandersson M. Detection of duplicate defect reports // Proceedings of the 29th international conference on Software Engineering / IEEE Computer Society. — 2007. — P. 499–510.

[24] Grinberg M. Flask web development: developing web applications with python. — " O'Reilly Media, Inc.", 2018.

[25] Lee S., Kim H.-j. News keyword extraction for topic tracking // 2008 Fourth International Conference on Networked Computing / IEEE. — Vol. 2. — 2008. — P. 554–559.

[26] Hulth A. Improved automatic keyword extraction // Proceedings of the 2003 conference on Empirical methods in natural language processing / Association for Computational Linguistics. — 2003. — P. 216–223.

[27] Nugumanova A., Novosselov A. Automatic keywords extraction from the domain texts // 2013 International Conference on Current Trends in Information Technology / IEEE. — 2013. — P. 186–189.

[28] Keyword extraction by entropy difference / Zhen Yang, Jianjun Lei, Kefeng Fan, Yingxu Lai // Physica A: Statistical Mechanics and its Applications. — 2013. — Vol. 392, no. 19. — P. 4523–4531.

[29] Hahn U., Schulz S. Building a very large ontology from medical thesauri // Handbook on ontologies. — Springer, 2004. — P. 133–150.

[30] Lorge F., Pantland M. A., Rojahn T. Building an ontology by transforming complex triples. — 2014. — Jun. 10. — US Patent 8,747,115.

ДОДАТОК 1

Автоматичне виділення концептів та тез в онтологічно-орієнтованому порталі

Копії публікацій

УКР.НТУУ “КПІ ім. Ігоря Сікорського”.ТІ41160_18М

Аркушів 1

Київ 2019

Ministry of Education and Science of Ukraine

National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Heat and Power Engineering Faculty

Glushkov Institute of Cybernetics, National Academy of Sciences of Ukraine

Polytechnic Institute of Tomar (Portugal)

Smart Cities Research Center (Portugal)

MODERN ASPECTS OF SOFTWARE DEVELOPMENT

Proceedings of
VI International Scientific and Practical Virtual Conference of
Software Development Specialists

June, 24, 2019

Kyiv, Ukraine

Modern Aspects of Software Development: Proceedings of VI International Scientific and Practical Virtual Conference of Software Development Specialists, June, 24 2019 p. – Kyiv: Igor Sikorsky KPI, 2019. – 189 p.

The Proceedings of VI International Scientific and Practical Virtual Conference of Software Development Specialists "Modern Aspects of Software Development" consists of the results of scientific researches and practical developments in the following directions: software engineering, computer ecological-economic monitoring, computerized design

Text is printed in the authors' edition.

ISBN

CONTENT